Pós-Graduação em Ciência da Computação

Sandrine Ventura Martins

SELF: Um framework conceitual de apoio ao ensino da Engenharia de Software

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
http://cin.ufpe.br/~posgraduacao

Recife
2019

Sandrine Ventura Martins

SELF: Um framework conceitual de apoio ao ensino da Engenharia de Software

Trabalho apresentado ao Programa de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Engenharia de Software e Linguagens de Programação
Orientador: Vinicius Cardoso Garcia

Recife
2019

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

M386s Martins, Sandrine Ventura
SELF: um framework conceitual de apoio ao ensino de engenharia de software / Sandrine Ventura Martins. – 2019.
136 f.: il., fig., tab.

Orientador: Vinicius Cardoso Garcia.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2019.
Inclui referências e apêndices.

1. Engenharia de software. 2. Educação. I. Garcia, Vinicius Cardoso (orientador). II. Título.

005.1 CDD (23. ed.) UFPE - CCEN 2020 - 22

Sandrine Ventura Martins

“SELF: Um framework conceitual de apoio ao ensino de Engenharia de Software”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovado em: 26 de agosto de 2019.

BANCA EXAMINADORA

_____

Profa. Dra. Simone Cristiane dos Santos
Centro de Informática / UFPE

_____

Profa. Dra. Ariane Nunes Rodrigues
Faculdade de Ciências e Tecnologia de Garanhuns/UPE

_____

Prof. Dr. Vinicius Cardoso Garcia
Centro de Informática / UFPE
(**Orientador**)

AGRADECIMENTOS

Agradeço primeiramente a Deus, reconhecendo que tudo é dEle, por Ele e para Ele. Em especial agradeço aos meus pais, Etiene Martins e Samuel Martins, e aos meus irmãos, Maxwell Martins e Suelen Martins por nunca medirem esforços para me apoiar e ajudar em cada etapa da minha vida, e sempre estarem ao meu lado, mesmo que a distância nos separe fisicamente.

Agradeço também a William Lins, que não me deixou enfraquecer nos últimos momentos de escrita desta dissertação me lembrando de que eu sou mais forte do que imagino. Te amo!

A todos os professores que me acompanharam por esta jornada e me ensinaram mais que os conteúdos de suas disciplinas, me ensinaram a ser uma pessoa e uma profissional excelente. Em especial ao professor Vinicius Garcia pelo interesse, confiança, empolgação e tempo dedicado a realização deste trabalho.

Por fim, agradeço aos meus amigos, em especial Yasmine Santos, Beatriz Arruda e Rayelle Vera Cruz pelas palavras de incentivo ao longo desses 2 anos.

RESUMO

Esta pesquisa de mestrado está firmada no contexto do ensino de Engenharia de Software (ES), tendo como motivação a necessidade de desenvolver competências técnicas e não técnicas nos alunos que pretendem atuar nessa área. A educação em Engenharia de Software, ao longo dos anos, tem incorporado novas abordagens e técnicas visando a melhoria na qualidade do seu processo de ensino-aprendizagem. Essa mudança reflete a necessidade de se mudar as metodologias formais usadas em sala em de aula, o foco precisa ser tanto na teoria quanto na aprendizagem experiencial. Assim, neste trabalho, é proposto um framework composto por etapas baseadas no ciclo PDCA (Plan – Do – Check – Act) para auxiliar o professor na construção de uma disciplina que leve em consideração o objetivo da disciplina e o aspecto prático da Engenharia de Software, utilizando as principais características do PBL (*Problem Based Learning*) na realização de um projeto. O *framework* foi aplicado em contexto real e seus resultados serão mostrados e discutidos. A avaliação foi feita através de um painel de especialistas e destaca que o *framework* permite que o professor conduza o processo de ensino de forma mais clara, fazendo com que o aprendizado, que ocorre no aluno, seja mais objetivo.

Palavras-chaves: Educação. Engenharia de Software. Framework. PBL.

ABSTRACT

This master's research is signed in the context of the teaching of Software Engineering (ES), having as motivation the need to develop technical and non-technical competencies in the students who intend to act in this area.The education in Software Engineering, over the years, has incorporated new approaches and techniques aimed at improving the quality of its teaching-learning process.This change reflects the need to change the formal methodologies used in class classroom, the focus needs to be both in theory and experiential learning. Thus, in this work, a framework consisting of steps based on the PDCA cycle (Plan – Do – Check – Act) is proposed to assist the teacher in the construction of a discipline that takes into consideration the objective of the discipline and the practical aspect of Software engineering, Using the main characteristics of the PBL (Problem Based Learning) in the realization of a project. The framework was applied in real context and its results will be shown and discussed. The evaluation was made through a panel of experts and highlights that the *framework* allows the teacher to conduct the teaching process more clearly, making learning, which occurs in the student, to be more objective.

Keywords: Education. Software Engineering. Framework. PBL.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Este capítulo está organizado em seis seções. A Seção 1.1 descreve o contexto ao qual a pesquisa está vinculada. A Seção 1.2 apresenta a motivação para o desenvolvimento deste trabalho. A seção 1.3 apresentam argumentos que justificam sua relevância para o contexto, além de motivos para a realização da pesquisa. A caracterização do problema é descrita na Seção 1.4 e os objetivos são apresentados na Seção 1.5. Por fim, a Seção 1.6 apresenta a estrutura deste trabalho.

1.1 CONTEXTO

Para (Nauman; Uzair, 2007), as principais técnicas de ES são ensinadas pela academia, porém, aos alunos não são expostos as dificuldades encontradas na realidade das empresas e, por isso, sugerem que haja uma simulação do ambiente da indústria na sala de aula. Segundo (ANTUNES, 2011) durante muito tempo confundiu-se "ensinar" com "transmitir", desta forma o aluno era apenas o agente passivo da aprendizagem enquanto o professor era um transmissor do conhecimento. Sendo que estudos como (GRILLO, 2002; JUNIOR; TANI; MANOEL, 2008) sugerem que o ensino no qual o aluno participa mais ativamente das aulas tende a ser mais eficaz. Os estudos de (BECKMAN; ALONIS, 1997; GIBBS; JR; GIBBS, 1994) mostram que a qualidade dos profissionais de ES, e portanto do desenvolvimento de softwares, é baseada principalmente na qualidade do ensino destes profissionais. Sendo assim, para melhorar as dificuldades encontradas para se desenvolver um software é importante melhorar a qualidade na educação das disciplinas que abordam essa temática.

Há um entendimento entre os docentes de disciplinas de informática de que é necessário trabalhos práticos para que os alunos consigam assimilar o conhecimento transferido (TOMAYKO, 1987; SCHNEIDER; JOHNSTON, 2003; STEVENS, 2001). No caso da disciplina ES estes estudos de caso normalmente são realizados de maneira a enfatizar a aprendizagem de aspectos tecnológicos tais como linguagens de especificação, métodos e ferramentas de apoio. Desta forma, os aspectos sociais do desenvolvimento, tais como cooperação e comunicação entre os desenvolvedores, deixam de ser trabalhados (HAZZAN; DUBINSKY, 2003).

O ensino de ES utilizando projetos práticos era a principal abordagem utilizada, mas nos últimos dez anos, abordagens alternativas vêm sendo propostas e adotadas (MARQUES; QUISPE; OCHOA, 2014). Por isso, não existe atualmente uma abordagem principal para conduzir experiências práticas (MALIK; ZAFAR et al., 2012), o que existem são abordagens mais ou menos efetivas, dependendo do contexto de ensino.

A maioria desses estudos salienta a necessidade de fornecer ao aluno a oportunidade de realizar atividades práticas, mas não especifica como o professor pode criar sua disci-

plina para que essas atividades sejam trabalhadas de forma eficiente, dando ao aluno as competências desejadas. Apesar de haver várias propostas relatadas na literatura, não há uma solução clara para lidar com os desafios encontrados no ensino de ES (MARQUES; QUISPE; OCHOA, 2014).

Neste contexto, a pesquisa enfatiza a utilidade de um processo com etapas e atividades bem definidas para estruturar o trabalho do professor, ao representar ações essenciais: 1) planejamento da disciplina a fim de definir objetivos claros e precisos, que reflitam competências que serão trabalhadas nos alunos; 2) a execução do processo de ensino orientado a atividades práticas que permitem preparar o aluno para situações que enfrentarão no mercado de trabalho; 3) acompanhamento dos instrumentos de avaliação para certificar o alcance dos objetivos e; 4) realização de melhorias ao longo do processo de ensino-aprendizagem por meio de *feedbacks* contínuos.

Neste trabalho, optou-se por adotar as abordagens práticas relatadas na literatura que são consideradas humanistas, pois permitem um ensino mais centrado no aluno, como também a realização de projetos práticos e a adoção de Problem-Based Learning (PBL) que desenvolvem a habilidade dos alunos trabalharem em equipes para resolverem problemas, o que permite prepará-los melhor para situações que enfrentarão no mercado de trabalho (SANTOS; ALEXANDRE; RODRIGUES, 2015).

1.2 MOTIVAÇÃO

Segundo (PIAGET; BUEY et al., 1972), a educação permanente e a renovação incessante do conhecimento são extremamente necessárias, no entanto, o essencial não está somente em aprender, mas sim em aprender a aprender. A ideia proposta por (PIAGET; BUEY et al., 1972), afirma que o que caracteriza a aprendizagem é o movimento de um saber fazer a um saber. Neste contexto, é enfatizada a necessidade de que a relação pedagógica entre professor e aluno seja elaborada com base metodológica e planejamento adequado. O professor deve ser responsável por um esforço construtivo, agrupando teorias modernas de aprendizagem.

Para (GIBBS; JR; GIBBS, 1994), a qualidade do ensino e a compreensão dos conteúdos e temas da engenharia de software têm grande impacto na melhoria do processo de desenvolvimento de software e, consequentemente, podem apoiar na busca por soluções de problemas e deficiências observadas na prática, no contexto da indústria de software. Por isso, por ser uma importante disciplina da área de Ciência da Computação, muitos defendem a aplicação prática das teorias estudadas de modo a enfatizar a aprendizagem de aspectos tecnológicos, tais como linguagens de especificação, métodos e ferramentas e também os aspectos sociais do desenvolvimento, como a cooperação e a comunicação entre os desenvolvedores (SILVA; LEITE; KOOGAN, 2004).

Para que o ensino de ES esteja ao nível do cenário atual de desenvolvimento de software, é necessário que este inclua habilidades e atitudes que vão além de conceitos, méto-

dos e técnicas (BRITO et al., 2018; NASCIMENTO et al., 2013). Isto implica que o processo de ensino puramente tradicional seja adaptado para dar mais atenção à complexidade das interações e de como se dá a construção de software colaborativamente, em um ambiente real (PINTO et al., 2017).

A engenharia de software (ES), de forma ampla, é a área do conhecimento que fornece toda a estrutura para o desenvolvimento e manutenção de software (CURRICULA, 2004). Esta área está preocupada principalmente com a aplicação de teoria, conhecimento e prática para o desenvolvimento efetivo e eficiente de sistemas de software, visando a implementação de produtos de qualidade, econômicos, úteis e no prazo esperado, cujos resultados devem satisfazer os requisitos dos usuários (CURRICULA, 2008). Em relação a ES como disciplina, (SOMMERVILLE, 2011) afirma que ela é uma disciplina da computação relacionada com a produção de software de alta qualidade, respeitando os requisitos e as restrições de tempo e custo.

A formação de profissionais qualificados na área de ES está ligada a qualidade da educação que esses profissionais receberam enquanto alunos, que deve incluir além de conhecimentos básicos na área de computação, o ensino de conceitos, processos e técnicas para definição, desenvolvimento e manutenção de software (CURRICULA, 2008). A praticidade é uma das características e requisitos representativos da educação em ES. Assim, é importante que seu ensino possa ser personalizado para refletir as necessidades da indústria

1.3 JUSTIFICATIVA

Para (MEIRA, 2015) é quase uma irresponsabilidade que os futuros profissionais de ES façam os cursos que fazem hoje, e que saiam deles, em quase qualquer instituição, tão ignorantes da vida real. Deve-se resolver o problema de onde e como aprender ES apropriadamente, pois a incompetência dessa formação é um duplo problema econômico: recursos investidos no ensino dessas competências e habilidades se perdem em sua quase totalidade e, por causa disso, são as empresas que têm, na prática, que investir para formar seu capital humano (MEIRA, 2015).

A formação qualificada e a capacitação de profissionais são cada vez mais necessárias na sociedade em que vivemos. Seja em cursos de curta duração, graduação ou pós-graduação, formar bons profissionais faz parte do compromisso das Instituições de Ensino Superior (IES) (PORTELA; VASCONCELOS; OLIVEIRA, 2016). Especificamente no ensino de ES, a qualidade dos profissionais está diretamente relacionada à qualidade da educação (PRIKLADNICKI et al., 2009). A fim de atingir essa qualidade, a educação e a capacitação de profissionais de software devem incluir não apenas conhecimentos básicos na área de Computação, como também o ensino de conceitos, processos e técnicas para definição, desenvolvimento e manutenção de software (CURRICULA; SOCIETY, 2013).

No cenário atual, além de abordar o conteúdo, é preciso que os professores tratem de aspectos didáticos e pedagógicos no ensino (ERICONE, 2001). Trabalhar em problemas reais é um elemento fundamental para o desenvolvimento de habilidades técnicas e não técnicas nos alunos. Essa experiência desenvolve simultaneamente o conhecimento técnico encontrado na gestão de projetos, juntamente com as competências não-técnicas de forma integrada e muda o estilo de ensino, para ser centrado no aluno, que devem participar ativamente no processo de aprendizagem, enquanto o professor assume o papel de facilitador (GONZÁLEZ-MORALES; ANTONIO; GARCÍA, 2011).

1.4 CARACTERIZAÇÃO DO PROBLEMA

Como citado anteriormente, a educação em ES é confrontada com inúmeros desafios. Apesar de existirem guias para o ensino de ES que fornecem orientações gerais sobre conteúdo curricular, sequenciamento e pedagogia, eles não fornecem detalhes sobre como implementar e entregar esses currículos.

Um dos maiores desafios, mesmo para disciplinas de engenharia mais maduras, é o problema de como preparar os alunos para a prática profissional em um ambiente acadêmico. Segundo (Meyer, 2001), a academia não deve assumir toda essa responsabilidade, pois a universidade não é uma empresa, mas precisa preparar os estudantes para os reais desafios que enfrentarão.

Sendo assim, diversos autores (GIMENES, 2015; HILBURN; TOWHIDNEJAD, 2007; GARG; VARMA, 2009) destacam dilemas didáticos envolvidos na disciplina de ES como: práticas conteudistas e carga horária insuficiente, dificuldade em trazer experiências práticas, e pouco interesse dos alunos por conta de aulas puramente teóricas. A partir das dificuldades expostas nesta seção, é possível destacar algumas lacunas principais na área de ensino de ES:

1. Ensino em sala de aula não é efetivo: muitos professores usam a pedagogia tradicional baseada em palestras didáticas. Este tipo de pedagogia é adequado para transmitir conhecimentos teóricos, mas ES é uma disciplina de engenharia e deve se concentrar na aplicação de conhecimentos e habilidades e outros aspectos práticos do desenvolvimento de software.

2. Restrições de tempo: ES é uma disciplina de um semestre na maioria dos programas de computação. Isso geralmente significa 2-3 horas de aula por semana, por cerca de 16-18 semanas, de forma eficaz. Simplesmente não há tempo suficiente para discussão aprofundada de todos os seus tópicos.

3. Falta de habilidades não técnicas: Existe uma grande dificuldade no desenvolvimento de competências essenciais, como habilidades para resolver problemas, auto-

didatismo, habilidades de comunicação, etc. Essas habilidades são imprescindíveis para engenheiros de software (GIMENES, 2015).

Outra questão principal que se enfrenta na ES é aprender estudando (na faculdade) vs aprendendo fazendo (no trabalho). Em (Ghezzi; Mandrioli, 2005) os autores acreditam fortemente que as duas formas de aprendizado são necessariamente diferentes e complementares. No entanto, seria um erro fatal adotar a ideia de que os dois mundos deveriam ignorar — ou até mesmo depreciar — uns aos outros. A aprendizagem no nível acadêmico deve munir os alunos com as habilidades imprescindíveis que lhes permitirão prosseguir para a aprendizagem ao longo da vida no trabalho. Em um campo contínuo e em rápida evolução, como o software, a educação deve enfatizar os princípios e reconhecer quais são os conceitos estáveis e duradouros da disciplina (Ghezzi; Mandrioli, 2005).

Além do desafio de se ensinar princípios de ES que podem não ser diretamente aplicáveis, há também a dificuldade de se integrar o ensino com atividades práticas. Se por um lado apenas estudar princípios em livros didáticos e até mesmo fazer exercícios não é "aprendizado"real sem uma experiência prática. Por outro lado, pode ser impossível imitar a complexidade dos projetos da vida real em um ambiente educacional. Na verdade, muitos detalhes devem ser abstraídos para tornar os problemas gerenciáveis dentro dos termos típicos do ambiente acadêmico (Ghezzi; Mandrioli, 2005).

Essas limitações acabam por fazer que os alunos não vivenciem as oportunidades de aprendizados e, consequentemente, não desenvolvem as competências esperadas para um profissional que pretende atuar na área de ES. Por isso surge a necessidade de instruir o profissional docente em conciliar de forma efetiva as atividades focadas no professor e as focadas no aluno na disciplina em um curto prazo de tempo e que possa ser aplicado com instrumentos educacionais de fácil alcance para as universidades. A fim de tratar as problemáticas identificadas e ajudar a diminuir as lacunas da área de ensino de ES, a principal questão de pesquisa investigada por esta tese é:

Questão de Pesquisa (QP):Como professores de Engenharia de Software podem construir uma disciplina que desenvolva competências técnicas e não técnicas nos alunos?

Com o intuito de refinar a investigação sob este aspecto, esta tese considera as seguintes questões:

Q1. Quais os desafios que um professor pode se deparar ao ensinar uma disciplina de ES?

Q2. Quais soluções existentes podem contribuir para o ensino de ES, de forma a reduzir os impactos dos desafios?

Q3. Quais as características desejáveis de uma solução que dê suporte ao ensino de competências técnicas e não técnicas em ES?

1.5 OBJETIVOS

O principal objetivo da pesquisa e conceber e avaliar uma estrutura conceitual para o ensino da disciplina de ES, afim de desenvolver competencias tecnicas de forma prática na area. Um framework conceitual pode ser definido como uma estrutura usada para entender o contexto e informar a direção de um projeto de pesquisa, usando suas informações para orientar o planejamento deste projeto (MAGHER, 2016). Uma estrutura conceitual também usa pesquisas anteriores para determinar uma teoria e metodologia para um projeto de pesquisa atual. Frameworks conceituais são produtos gerados de processos qualitativos que reúnem conceitos para fornecer a compreensão abrangente de um fenômeno (JABAREEN, 2009). Com base nos problemas identificado na Seção 1.2 e no objetivo geral desta pesquisa, têm-se os seguintes objetivos específicos:

- Analisar os principais desafios para o ensino de ES
- Identificar a importância das diversas áreas da ES para alunos, professores e profissionais
- Identificar e entender as atuais abordagens de ensino focadas no aluno
- Estruturar uma proposta de ensino da disciplina de ES, que possa combater alguns dos desafios identificados no estudo

1.6 ESTRUTURA DO TRABALHO

Além deste capítulo introdutório, este trabalho está organizado da seguinte forma:

- O capítulo 2 descreve a fundamentação teórica mediante análise de diretrizes curriculares e processo de ensino-aprendizagem;
- O capítulo 3 descreve a metodologia de pesquisa utilizada nesta tese;
- O capítulo 4 apresenta a proposta deste trabalho, por meio do SELF - Framework de apoio ao ensino de ES;
- O capítulo 5 apresenta a aplicação e avaliação do SELF; e
- O capítulo 6 traz as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta as construções teóricas usadas como base para a formulação deste trabalho. Na Seção 2.1, é apresentado o guia para o corpo de conhecimento de ES que será núcleo central do entendimento de ES para este trabalho. Na Seção 2.2, é feita uma análise das diretrizes curriculares mais relevantes com foco em ES. Como o *framework* que será proposto mais a frente neste trabalho será aplicado no curso de Sistemas de Informação, serão apresentadas as últimas diretrizes para este curso. Na Seção 2.3 é feita uma discussão acerca dos currículos apresentados. Nas Seções 2.4, 2.5 e 2.6 uma caracterização pedagógica do processo de ensino-aprendizagem é apresentada. Na Seção 2.7 é apresentado o kaizen, uma filosofia de melhoria contínua. Na Seção 2.8 é mostrado o diagrama utilizado para estruturar o *framework*. Por fim, a Seção 2.9 traz as conclusões do capítulo.

2.1 SWEBOK

O Guide to the Software Engineering Body of Knowledge (SWEBOK) (BOURQUE; FAIRLEY et al., 2014) é uma iniciativa da IEEE Computer Society que tem o propósito de criar um consenso sobre as áreas de conhecimento da ES e seu escopo. Começou como uma colaboração entre IEEE CS, ACM e a Université du Québec à Montréal e contou com a participação internacional da indústria, sociedades de profissionais, academia e autores. Para sua versão atual, V3.0, os editores receberam e responderam aos comentários de aproximadamente 150 revisores de 33 países.

O SWEBOK fornece uma referência importante para o estabelecimento da ES como uma disciplina reconhecida de engenharia; promove uma visão consistente da ES em todo o mundo; caracteriza o conteúdo da disciplina de ES; e serve como base para o desenvolvimento curricular de ES. O SWEBOK expõe a amplitude e diversidade da área de ES, o que é um importante passo para aperfeiçoar processos e capacitar pessoas, levando ao aprimoramento da gestão de projetos e ao enriquecimento do seu produto final, por isso, ele será visto neste trabalho como sendo o núcleo central do entendimento de ES.

O SWEBOK é dividido em 15 áreas de conhecimento (KA) para ES e 7 disciplinas que se cruzam com a ES. Suas áreas de conhecimento com uma breve descrição das mesmas serão listadas a seguir:

- Requisitos de Software: os requisitos de um software expressam as necessidades de um software para que ele atenda um determinado propósito. Os requisitos podem tanto detalhar o comportamento e funções do software, quanto também suas restrições e limitações. A área de requisitos de software engloba todo o processo envolvendo a coleta (elicitação), análise, especificação e validação dos requisitos.

Esta área é fundamental para que o software tenha um resultado esperado, pois todo o projeto é desenvolvido com base nos requisitos coletados e validados, caso esses requisitos não estejam em conformidade com a real necessidade, pode acontecer da construção de um produto que não resolva o problema proposto. Esta área também trata da classificação dos requisitos em grupos, para simplificar a análise e rastreamento dos mesmos;

- Design de Software: esta área de conhecimento engloba a estrutura arquitetural do software, os componentes, módulos, interfaces, etc, com base nos requisitos coletados e validados, pois os requisitos são o mapa para a construção de uma arquitetura que satisfaça a necessidade do produto a ser desenvolvido;

- Construção de Software: a implementação do software está dentro desta área, também faz parte desta área os testes de unidade, de integração, o que torna ela extremamente relacionada com as áreas de Design e Testes. As questões de linguagem, codificação, complexidade, reutilização, integração, qualidade, dentre outras também são tratadas dentro desta área;

- Teste de Software: o teste de software consiste na verificação dinâmica, para avaliar se um determinado programa se comporta de forma esperada, em um escopo finito de casos de teste, correspondentes aos requisitos funcionais e não funcionais, definidos durante a modelagem do sistema;

- Manutenção de Software: após a finalização de um projeto de software, se o mesmo continuar operante, naturalmente ele sofrerá mudanças e necessitará evoluir. Dessa forma, defeitos serão descobertos, operações de mudança de ambiente e novas requisições dos usuários. Assim, a manutenção do ciclo de vida do software se dá, desde a primeira implementação de uma funcionalidade até o dia em que o mesmo deixa de ser utilizado, garantindo assim o software satisfaça as necessidades de seus usuários;

- Gerenciamento de Configuração de Software: a gerência de configuração de software, é responsável por identificar a configuração de um sistema, sobre diferentes perspectivas no decorrer do tempo, com o propósito de controlar sistematicamente as mudanças nos processos de configuração e manutenção, garantindo a integridade e rastreabilidade ao longo do ciclo de vida do software/sistema;

- Gerenciamento da Engenharia de Software: a engenharia de software contém partes de gerência de projeto em seu escopo para assegurar que os serviços de engenharia de software são entregues de forma eficiente e que beneficiam as partes interessadas. Essa área de conhecimento da engenharia de software abrange tanto no nível organizacional, gerência de projeto e gerência do programa de métricas;

- Processos de Engenharia de Software: é um conjunto de atividades que transformam entradas em saídas. Muitos dos processos das engenharias tradicionais consistem em transformar artefatos físicos em outras coisas mas a engenharia de software transforma ideias em produtos de software;
- Modelos e Métodos de Engenharia de Software: têm o objetivo de tornar as atividades sistemáticas, repetíveis e orientadas ao sucesso. Modelagens e ferramentas são usadas para ajudar o engenheiro de software a criar um produto apropriado às partes interessadas;
- Qualidade de Software: refere-se às características desejáveis de produtos de software e aos processos, ferramentas e técnicas utilizadas para alcançar essas características;
- Prática Profissional em Engenharia de Software: está relacionada com os conhecimentos, habilidades e atitudes que engenheiros de software devem possuir para a prática da engenharia de software de uma forma profissional, responsável e ética;
- Economia de Engenharia de Software: tem como objetivo abordar a Engenharia de Software em um contexto de negócios, ou seja, em um contexto econômico. O sucesso de um software, serviço e solução dependem de uma boa gerência de custos em Engenharia de Software;
- Fundamentos de Computação: abrange o desenvolvimento e o ambiente operacional em que o software evolui é executado;
- Fundamentos de Matemática: está preocupada com o entendimento dos conceitos básicos de lógica entre os engenheiros de software, que por sua vez será traduzida em código de linguagem de programação;
- Fundamentos de Engenharia: está preocupada com os fundamentos de engenharia que se aplicam a engenharia de software e outras disciplinas de engenharia.

Apesar de ser um dos principais documentos para fornecer fundamentos para o desenvolvimento curricular em ES, o SWEBOK apresenta alguns problemas quando se diz respeito a sua estrutura para as áreas de conhecimento. As áreas de conhecimento não possuem uma estrutura comum entre si, por exemplo: a área de Design de Software possui conceitos relacionados a área, mas a área de requisitos não. Os padrões foram criados por grupos de voluntários separados que se concentraram em diferentes áreas do conhecimento. Provavelmente, os autores não conseguiram chegar a um consenso comum que resultasse em uma estrutura comum (KAJKO-MATTSSON; SJÖGREN; LINDBÄCK, 2017). Outro problema aborda as dificuldades de avaliação dos programas educativos. Os corpos de conhecimentos para as áreas individuais não oferecem nenhuma relação para avaliar se apenas um tipo de conhecimento foi abordado, por exemplo: como se avalia o REQ

2.4(Qualidade e melhoria do processo) quando apenas a qualidade do processo é implementada no programa educacional? (KAJKO-MATTSSON; SJÖGREN; LINDBÄCK, 2017)

2.2 DIRETRIZES CURRICULARES NACIONAIS E INTERNACIONAIS

2.2.1 Sociedade Brasileira de Computação

Em julho de 1999, a Assembléia Geral da Sociedade Brasileira de Computação (SBC) aprovou uma proposta de currículo de referência para os cursos de graduação na área de Computação e Informática, referenciado como CR99. Após a aprovação, as revisões foram separadas entre os cursos de sistemas de informação, que teve sua revisão feita em 2003, e os cursos de ciência da computação e engenharia da computação com revisão feita no ano de 2005.

Já em 2017, a SBC apresentou o livro 'Referenciais de Formação para os Cursos de Graduação em Computação' (RFs) que contém referenciais de formação para os cursos de Ciência da Computação, Engenharia de Computação, Engenharia de Software, Licenciatura em Computação, Sistemas de Informação e Cursos Superiores de Tecnologia em Computação. Os RFs não são currículos, mas sim um material de consulta para quem estiver elaborando seus currículos (ZORZO et al., 2017).

Os primeiros currículos de referência propostos pela SBC definem o perfil do egresso (ou do profissional) com uma forte caracterização técnica e também ética-social, os RFs, por outro lado, em consonância com as Diretrizes Nacionais Curriculares homologadas em 2016 (ZORZO et al., 2017) , estão construídos a partir da noção de competência, seguindo a visão atualmente mais recomendada para estruturação de currículos, métodos de ensino e aprendizagem, e métodos de avaliação (ZORZO et al., 2017).

Essa noção de competências, traz diversas vantagens como: capacidade em dar significado aos conteúdos de conhecimento que compõem o currículo; a ampliação do currículo para incluir habilidades e atitudes, além de conhecimento; e uma maior aderência ao perfil do egresso esperado pelo curso (KLINK; BOON; SCHLUSMANS, 2007).

- Sistemas de Informação:

  O Referencial de Formação para os Cursos de Bacharelado em Sistemas de Informação (RF-SI) (ZORZO et al., 2017) é estruturado, em linhas gerais da seguinte forma: o perfil esperado para o egresso determina o objetivo geral do curso, decomposto em diferentes eixos de formação. Eixos de formação objetivam capacitar o egresso em competências genéricas. Para que o egresso possa se apropriar destas competências genéricas, é necessário que este possa desenvolver competências derivadas que requerem a mobilização de conteúdos específicos ministrados em unidades curriculares ou disciplinas.

Para ES, os conteúdos estão listados no eixo de formação 'Desenvolvimento de Software para Sistemas de Informação'. Este eixo tem como competência geral: 'Gerenciar os sistemas de informação em contextos sociais e organizacionais, avaliando as necessidades de informatização nestes sistemas, especificando soluções de software para sistemas de informação, produzindo o software para o atendimento destas necessidades, aplicando processos, técnicas e ferramentas de desenvolvimento de software, implantando o software em contextos sociais e organizacionais de sistemas de informação, mantendo sua operação e avaliando o impacto de seu uso.

Os conteúdos de ES que podem ser vistos neste eixo são: Análise de Riscos, Engenharia de Requisitos, Gerência de Projetos, Qualidade de Software , Gerência de Configuração de Software, Arquitetura de Software, Verificação e Validação de Software , Teste de Software, Manutenção de software e Construção de Software.

O RF-SI traz benefícios quando se precisa determinar quais disciplinas e competências precisam estar presentes no projeto pedagógico do curso, pois a falta de alguma dessas competências pode levar a instituição de ensino a não formar o perfil desejado do egresso em Sistemas de Informação. Uma vez determinado pelo curso quais eixos e competências serão abordadas com maior ou menor ênfase no curso, os conteúdos destas competências podem ser então articulados para a composição de disciplinas.

Apesar do RF-SI ser um documento que auxilia os coordenadores do curso de graduação a elaborar seus projetos pedagógicos, ele não traz ganhos maiores quanto ao auxílio aos professores para elaborarem sua grade curricular, pois para isso precisará ser levado em consideração diversos aspectos institucionais e regionais fazendo com que o RF-SI deva ser analisado com cautela. Os conteúdos propostos no RF-SI também não representam diretamente disciplinas da grade curricular, para que isso aconteça, eles precisam ser agregados.

2.2.2 Association for Computing Machinery (ACM)

Nas décadas desde os anos 1960 a Association for Computing Machinery (ACM) e a Computer Society of the Institute for Electrical and Electronic Engineers (IEEE-CS) produziram recomendações curriculares para enfrentar os desafios da rápida mudança da paisagem tecnológica da computação. Juntos, eles têm publicado diretrizes curriculares para as seguintes áreas: Ciência da Computação, Engenharia da Computação, Sistemas de Informação, Tecnologia da Informação e Engenharia de Software.

- Sistemas de Informação:

  A Diretriz Curricular IS2010 (TOPI et al., 2010) para Sistemas de Informação foi feita em parceria com a ACM e a Association for Information Systems (AIS). Esta diretriz especifica um curso geral (Fundamentos de Sistemas de Informação) para

fornecer uma compreensão e habilidades relacionadas aos Sistemas de Informação adequados para todos os alunos.

Esta diretriz está organizada em sete disciplinas básicas e um subconjunto de disciplinas eletivas. A maioria dos tópicos de ES estão listados na disciplina de Gerenciamento de Projeto de Sistemas de Informação. Esta disciplina abrange uma metodologia sistemática para a iniciação, planejamento, execução, controle e encerramento projetos. Seus tópicos são: introdução ao gerenciamento de projeto; gerenciamento de ciclo de vida; gerenciamento de equipes de projeto; gerenciamento de comunicação; iniciação e planejamento de projeto; gestão do escopo do projeto; gerenciamento de preparação de projeto; gerenciamento de recursos; qualidade do projeto; riscos do projeto; e gerenciamento da execução, controle e encerramento do projeto.

Como benefício, além de mostrar o currículo principal - que é flexível, adaptável e orienta as instituições de ensino a criarem sua grade curricular para cursos de Sistemas de Informação, formando assim, egressos competentes e adequados às responsabilidades do local de trabalho - a IS2010 ilustra seu uso em três contextos acadêmicos diferentes, com diferentes requisitos gerais de graduação.

No entanto, este currículo não evidencia conteúdos específicos de áreas que são importantes aos profissionais de sistemas de informação, como por exemplo a área de matemática. Mesmo que os profissionais da área não precisem do mesmo nível de profundidade matemática como muitos outros profissionais de computação, seria importante criar uma disciplina que traga conteúdos centrais de alta importância.

Um profissional que possui conhecimentos em matemática é capaz de resolver problemas e apresentar soluções claras, organizadas, criativas e eficientes. Em um ambiente em contínua mudança, é imprescindível que se tenha conhecimentos básicos em matemática, principalmente quando se trata de profissionais de tecnologia, tanto técnicos como voltados ao gerenciamento de projetos. Este pode ser o diferencial para os profissionais que pretendem se manter à frente, diante da quantidade de informações e dos avanços tecnológicos. Para compreender estes avanços, é preciso acompanhá-los, e para isso, a matemática se tornou uma aliada fundamental da computação.

2.3 DISCUSSÕES ACERCA DOS CURRÍCULOS

Na Tabela 1 é apresentado como as diretrizes curriculares apresentadas anteriormente tratam os pontos do SWEBOK:

Tabela 1 – Engenharia de Software nos currículos

SWEBOK	ACM	SBC
Requisitos de Software	É tratado na disciplina de "Análise e Design de Sistemas". O foco desta disciplina é na análise e documentação de requisitos de negócios, bem como na conversão desses requisitos em requisitos detalhados de sistemas e especificações de alto nível (por exemplo, mock-ups de formulários e relatórios).	É tratado no eixo de formação "desenvolvimento de Software para Sistemas de Informação". Onde os alunos aprenderão a especificar software para informatização de sistemas, elicitando os requisitos do software em conformidade com os requisitos do produto, dos processos e das partes interessadas, analisando e especificando seus requisitos funcionais e não funcionais e validando o seu potencial de solução das necessidades de sistemas de informação.
Design de Software	É tratado na disciplina de "Desenvolvimento de aplicações". O objetivo desta disciplina é apresentar aos alunos os conceitos e modelos fundamentais de desenvolvimento software para que eles possam entender os principais processos relacionados à criação de softwares funcionais.	É tratado no eixo de formação "desenvolvimento de Software para Sistemas de Informação". Onde os alunos aprenderão a projetar software para informatização de sistemas, determinando sua arquitetura, garantindo sua qualidade técnica e validando seu potencial de eficácia, eficiência, efetividade e sustentabilidade.
Construção de Software	É tratado na disciplina de "Desenvolvimento de aplicações". Os alunos aprenderão os conceitos básicos de projeto de software, estruturas de dados, programação, resolução de problemas e lógica de programação. Também irão desenvolver um software simples incorporando seu ciclo de vida.	É tratado no eixo de formação "desenvolvimento de Software para Sistemas de Informação". Onde os alunos aprenderão a construir software para informatização de sistemas avaliando sua qualidade técnica.
Teste de Software	É tratado na disciplina de "Desenvolvimento de aplicações" e na disciplina de "Introdução a interação Humano-Computador". Na primeira disciplina os alunos aprenderão a criar testes unitários de suas aplicações e na segunda disciplina aprenderão sobre testes de usabilidade.	É tratado no eixo de formação "desenvolvimento de Software para Sistemas de Informação". Onde os alunos aprenderão a testar o funcionamento do software desenvolvido, e validando seu atendimento às necessidades de eficácia, eficiência, efetividade e sustentabilidade.

Continua na próxima página

Tabela 1 – Continuação

SWEBOK	ACM	SBC
Manutenção de Software	É tratado na disciplina "Auditoria e Controles". Os alunos aprenderão sobre atividades de manutenção.	É tratado no eixo de formação "desenvolvimento de Software para Sistemas de Informação". Onde os alunos aprenderão a. manter software, corrigindo falhas, adaptando-o ao seu contexto, identificando e implementando melhorias, migrando softwares legados e retirando software.
Gerenciamento de Configuração de Software	É tratado na disciplina "Gerenciamento de Projetos de SI". Os alunos aprenderão sobre controle de projetos através de rastreamento de informações e técnicas de controle de mudanças.	É tratado no eixo de formação "desenvolvimento de Software para Sistemas de Informação". Onde os alunos aprenderão a gerenciar projetos de produção de software para informatizar sistemas aplicando processos, técnicas e ferramentas de engenharia de software.
Gerenciamento da Engenharia de Software	É tratado na disciplina "Gerenciamento de Projetos de SI". Os alunos aprenderão sobre o gerenciamento de projetos e terão uma visão do mesmo como uma atividade complexa, onde vários tipos de tecnologia (incluindo software de gerenciamento de projetos, bem como software para apoiar a colaboração em grupo) são uma parte inerente do processo a este processo.	É tratado no eixo de formação "desenvolvimento de Software para Sistemas de Informação". Onde os alunos aprenderão a gerenciar projetos de produção de software para informatizar sistemas aplicando processos, técnicas e ferramentas de engenharia de software.
Processos de Engenharia de Software	É tratado nas disciplinas de "Desenvolvimento de aplicações" e "Análise e Design de Sistemas". Onde os alunos aprenderão sobre o ciclo de vida do desenvolvimento do software: reunindo requisitos, projetando, implementando e testando o software	É tratado no eixo de formação "Gestão de Sistemas de Informação e da Tecnologia da Informação". Onde os alunos aprenderão a gerenciar projetos de sistemas de informação e de tecnologia da informação para manutenção dos sistemas de informação organizacionais e da arquitetura de tecnologia da informação da organização, aplicando conceitos e processos de planejamento, acompanhamento e avaliação de projetos de tecnologia da informação.
Modelos e Métodos de Engenharia de Software	É tratado na disciplina "Gestão de processos de negócio". Os alunos serão introduzidos aos principais conceitos e abordagens para a gestão de processos de negócios e melhoria e ao uso de ferramentas básicas de modelagem de processos de negócios.	É tratado no eixo de formação "Gestão de Sistemas de Informação e da Tecnologia da Informação". Onde os alunos aprenderão a gerir a arquitetura de tecnologia da informação em organizações, identificando as demandas dos sistemas de informação organizacionais e elaborando soluções de tecnologia da informação para o seu apoio.

Continua na próxima página

Tabela 1 – Continuação

SWEBOK	ACM	SBC
Qualidade de Software	É tratado na disciplina "Gerenciamento de Projetos de SI". Onde os alunos aprenderão a gerenciar a qualidade do projeto, incluindo a identificação das ameaças à qualidade do projeto, técnicas para medir a qualidade do projeto, e as técnicas para garantir que a qualidade do projeto será alcançada.	É tratado no eixo de formação "desenvolvimento de Software para Sistemas de Informação". Onde os alunos aprenderão a validar o potencial de eficácia, eficiência, efetividade e sustentabilidade do software.
Prática Profissional em Engenharia de Software	É tratado na disciplina de "Fundamentos de Sistemas de Informação". Onde as expectativas e obrigações éticas profissionais serão revistas e a necessidade de habilidades de comunicação pessoal e interpessoal será discutida.	É tratado no eixo de formação "Desenvolvimento Pessoal e Profissional". Onde os alunos aprenderão a atuar profissionalmente planejando continuamente o seu desenvolvimento pessoal e profissional, contemplando os desafios pessoais, profissionais e da sociedade de forma proativa e crítica, agindo de acordo com princípios éticos profissionais que considerem o respeito aos direitos humanos, o compromisso com a sustentabilidade e responsabilidade socioambiental.
Economia de Engenharia de Software	É tratado nas disciplinas "Gerenciamento de Projetos de SI"; "Sistemas corporativos"; e "Fundamentos de Sistemas de Informação". Os alunos irão aprender sobre controle de custos em projetos; avaliar os custos e benefícios da implementação de um sistema corporativo; e compreender o valor dos investimentos em sistemas de informação, bem como aprender a formular um caso de negócios para um novo sistema de informação, incluindo a estimativa de custos e benefícios.	É tratado no eixo de formação "Visão Sistêmica". Onde os alunos aprenderão a elaborar soluções eficazes, eficientes, efetivas e sustentáveis de sistemas de informação, considerando aspectos tecnológicos, econômicos, sociais e ambientais

Continua na próxima página

Tabela 1 – Continuação

SWEBOK	ACM	SBC
Fundamentos de Computação	É abordado na disciplina de "Infraestrutura de TI". Onde os alunos aprenderão temas relacionados à arquitetura de computadores e sistemas e redes de comunicação, com foco geral nos serviços e capacidades que as soluções de infraestrutura de TI possibilitam em um contexto organizacional.	É tratado no eixo de formação "Infraestrutura para Sistemas de Informação". Onde os alunos aprenderão a avaliar a arquitetura física e lógica das redes de comunicação e de computadores para organização, utilizando conceitos dos modelos de referência, analisando a operação e desempenho de seus componentes, aplicando os conceitos de alta disponibilidade e balanceamento de carga, e utilizando máquinas virtuais e softwares de gerenciamento.
Fundamentos de Matemática	Não há uma disciplina que trate de conteúdos relacionados a esta área especificamente, mas o currículo ressalta que conhecimentos e habilidades nesta área são fundamentais.	É tratado no eixo de formação "Engenharia de Dados e Informações". Onde os alunos aprenderão a representar contextos do mundo real na forma de conjuntos, reconhecendo suas instâncias, analisando e estabelecendo relacionamentos entre conjuntos e definindo funções e relações aplicáveis a estes conjuntos; e interpretar fenômenos estatísticos, empregando-os em outras áreas do conhecimento.
Fundamentos de Engenharia	É abordado na disciplina de "Introdução a interação Humano-Computador". Onde os alunos serão apresentados as tecnologias contemporâneas utilizadas nos métodos de avaliação empírica.	Não há uma disciplina que trate de conteúdos relacionados a esta área.

Fonte: Elaborado pela autora (2019)

O desenvolvimento de um corpo de conhecimento da engenharia de software é uma grande realização, pois enfrenta o desafio de acompanhar a rápida mudança da paisagem da ES. Um corpo de conhecimento, como o SWEBOK, pode orientar no desenvolvimento de currículos de ES e serve como base para avaliar os conhecimentos e as competências inerentes a área (KAJKO-MATTSSON; SJÖGREN; LINDBÄCK, 2017). O SWEBOK e os currículos da ACM/IEEE e SBC ajudam a determinar quais pontos podem levar a criar uma estrutura curricular de sucesso e propõem projetos que levam a atacar possíveis fraquezas e melhorar a qualidade do ensino em cada área. Além disso, esses documentos determinam os elementos que devem ser adaptados, para que sejam efetivos não apenas nas suas áreas específicas, mas em outras áreas do conhecimento.

Os currículos da ACM/IEEE para os cursos de Ciência da Computação e Engenharia da Computação não foram explicitados na seção 2.2.2, mas tratam da ES como uma área que integra o seu corpo de conhecimento e que os estudantes podem aplicar o que será aprendido através da participação em um projeto, e este deve exigir que os estudantes trabalhem em equipe para desenvolver um sistema de software, e usem técnicas de ES eficazes para desenvolver e praticar suas habilidades de comunicação com as partes interessadas.

Os referenciais da SBC para Ciência da Computação e Engenharia da Computação abordam a ES de forma mais generalista, não há um detalhamento mais preciso de seus conteúdos, sendo estes listados apenas como “Engenharia de Software”. Nos eixos onde há conteúdos de ES as competências esperadas são “Desenvolver sistemas computacionais que atendam qualidade de processo e de produto, considerando princípios e boas práticas de engenharia de sistemas e engenharia de software” e “Criar, implementar e manter soluções computacionais eficientes para diversos tipos de problemas, envolvendo hardware, software e processos, analisando o espaço de projeto considerando restrições e custo-benefício; e criar e integrar componentes de hardware, de software e sua interface.” respectivamente.

Os currículos usam elementos acadêmicos, como a taxonomia de Bloom (BLOOM et al., 1956) para determinar os níveis de aprendizagem que se espera para cada competência esperada. Cada competência está associada aos níveis cognitivos da escala da Taxonomia de Bloom e cada curso pode traçar a melhor trajetória para alcançar estas competências usando os níveis mais básicos (Conhecer, Compreender e Aplicar) até aos mais altos (Criar, Avaliar e Analisar). Como a ES é tratada nesta pesquisa como uma disciplina, as competências esperadas dos alunos irão se basear nos níveis mais básicos.

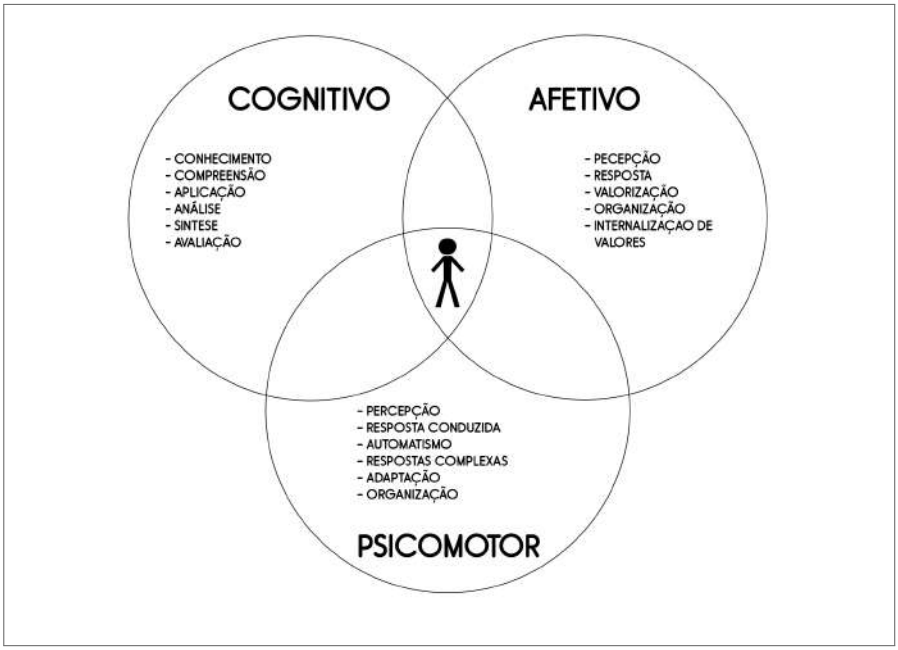
2.4 NÍVEIS DE APRENDIZAGEM

A taxonomia de Bloom é uma classificação dos diferentes objetivos e habilidades que os professores estabelecem para seus alunos (objetivos de aprendizado) (BLOOM et al., 1956). A classificação proposta por Bloom dividiu as possibilidades de aprendizagem em três grandes domínios:

- Cognitivo: relacionado ao aprender, dominar um conhecimento.
- Afetivo: relacionado a sentimentos e posturas.
- Psicomotor: relacionado a habilidades físicas específicas.

Cada um destes domínios tem diversos níveis de profundidade de aprendizado como apresentado na Figura 1. Por isso a classificação de Bloom é denominada hierarquia: cada nível é mais complexo e mais específico que o anterior.

Figura 1 – Taxonomia de Bloom



Fonte: Adaptado de (BLOOM et al., 1956).

Para especificar os níveis mais básicos de habilidades cognitivas, as diretrizes curriculares da ACM/IEEE para cursos de graduação em Engenharia de Software (BOURQUE; ROBERT,) categorizam uma terminologia baseada na taxonomia de Bloom (BLOOM et al., 1956) que consiste em conhecimento, compreensão e aplicação.

- Conhecimento: É definido como comportamentos e situações de teste que enfatizam o ato de lembrar. É esperado que o estudante armazene em sua mente determinadas informações, e o comportamento esperado mais tarde é recordar desta informação.
- Compreensão: É definida como o ato de entender o que está sendo apresentado. Quando os alunos são confrontados com uma comunicação, espera-se que eles saibam o que está sendo comunicado e sejam capazes de fazer algum uso do material ou ideias contidas nele. Envolve demonstrar uma compreensão de fatos e ideias organizando, comparando, traduzindo, interpretando, dando descrições e afirmando as principais ideias.

- Aplicação: Para aplicar algo requer que o aluno tenha compreensão do método, teoria, princípio ou abstração aplicada. Os alunos devem ser capazes de usar o conhecimento prévio para resolver problemas, identificar conexões e relacionamentos e como eles se aplicam em novas situações. Uma demonstração de "compreensão"mostra que o aluno pode usar a abstração quando seu uso é especificado. Uma demonstração de "aplicação"mostra que ele vai usá-lo corretamente, dada uma situação adequada em que nenhum modo de solução é especificado.

O uso da taxonomia pode ajudar a obter uma perspectiva sobre a ênfase dada a determinados comportamentos por um conjunto particular de objetivos educacionais. Assim, um professor, ao classificar os objetivos de um ciclo de ensino, pode descobrir que todos se enquadram na categoria taxonômica de conhecimento ou compreensão do conteúdo. Assim, o professor pode incluir, por exemplo, alguns objetivos que lidam com a aplicação deste conteúdo melhorando assim o seu processo de ensino-aprendizagem.

2.5 PROCESSO DE ENSINO-APRENDIZAGEM

Em relação ao processo de aprendizagem, (BORDENAVE; PEREIRA, 1985) afirmam que assim como o problema de um médico é conseguir que seus pacientes fiquem curados, o problema do professor é conseguir que seus alunos aprendam. Aprender é uma atividade que acontece no aluno e que é realizada pelo aluno.

Já em relação ao processo de ensino, existem diversos fatores que afetam este processo, relacionados com o tripé aluno-professor-assunto, e este processo consiste no manejo desses fatores e da sua dinamização em uma sequência mais ou menos planejada ou sistemática.

As abordagens do processo de ensino aprendizagem buscam reconhecer a dinâmica envolvida nos atos de ensinar e aprender, partindo do reconhecimento da evolução cognitiva do homem, e tentam explicar a relação entre o conhecimento pré-existente e o novo conhecimento (SANTORO; BORGES; SANTOS, 1999). A Tabela 2 apresenta as características básicas de algumas das principais abordagens:

Para o foco deste trabalho, daremos ênfase na abordagem humanista, pois ela possui as características que serão importantes na elaboração da proposta como foco no aluno e vê o professor como um facilitador da aprendizagem criando condições para que os alunos aprendam. Para entender melhor essa abordagem, serão apresentadas na Tabela 3 suas características em aspectos que serão relevantes para a proposta que será apresentada.

Para o processo de ensino-aprendizagem em ES no âmbito acadêmico, (BOURQUE; ROBERT,) ressalta que os estudantes podem melhor aprender a aplicar grande parte do material definido na área de conhecimento de engenharia de software através da participação em um projeto, que deve exigir que os estudantes trabalhem em equipe para desenvolver um software por meio do seu ciclo de vida, da forma como for possível. Apesar

Tabela 2 – Características básicas das abordagens de ensino-aprendizagem

Tradicional	**Comportamentalista**	**Humanista**	**Cognitiva**	**Sócio Cultural**
Busca conduzir o aluno ao contato com as grandes realizações da humanidade; Volta-se para o que é externo ao aluno: o programa, as disciplinas, o professor.	Denominada também como Behaviorista; O organismo está sujeito às contingências do meio; Origem empirista: o conhecimento é o resultado direto da experiência; O comportamento humano pode ser modelado e reforçado, através de recompensas e controle.	Enfoque no sujeito: principal elaborador do conhecimento humano; Ênfase às relações interpessoais, à vida psicológica e emocional, a personalidade do indivíduo em seus processos de construção pessoal da realidade - pessoa integrada	Estuda cientificamente a aprendizagem; Ênfase em processos cognitivos e na investigação científica. Na capacidade do aluno de integrar informações e processá-las; Abordagem interacionista.	Enfatiza aspectos sócio-político-culturais, preocupando-se com a cultura popular; Movimento de Cultura Popular: possibilitar uma real participação do povo como sujeito de um processo cultural.

Fonte: Adaptado de (SANTORO; BORGES; SANTOS, 1999)

da organização e execução de um projeto nesse âmbito ser um desafio, continua sendo a melhor maneira de aprender e aplicar os conhecimentos da engenharia de software.

Entretanto, abordagens alternativas podem ajudar os alunos a aprender de maneira mais efetiva, como por exemplo: a substituição de aulas expositivas por discussão de casos práticos (GNATZ et al., 2003), dinâmicas de grupo, uso de jogos (WANGENHEIM; SHULL, 2009) e *Capstone projects* (um esforço em grupo em que alunos executam um projeto do início ao fim) (CURRICULA, 2004).

A importância do processo de ensino-aprendizagem tem gerado a necessidade de o professor ter, não apenas conhecimentos sólidos na área em que pretende lecionar, mas também de habilidades pedagógicas suficientes para tornar o aprendizado mais eficaz.

A partir deste cenário, muitos(as) professores(as) utilizam abordagens centradas no aluno como metodologia de ensino:

- *Problem Based Learning*: É uma abordagem instrucional (e curricular) centrada no aluno que capacita os alunos a realizar pesquisas, integrar teoria e prática, e aplicar conhecimentos e habilidades para desenvolver uma solução viável para um problema definido (SAVERY, 2015).

- *Challenge Based Learning*: A aprendizagem baseada em desafios é uma experiência de aprendizado colaborativa na qual professores e alunos trabalham juntos para aprender sobre questões convincentes, propor soluções para problemas reais e agir.A abordagem pede aos alunos que reflitam sobre sua aprendizagem e o impacto de suas ações e publiquem suas soluções para um público mundial (NICHOLS; CATOR, 2008)

- *Project Based Learning*: O aprendizado baseado em projeto é um modelo que organiza a aprendizagem em torno de projetos. Os projetos são tarefas complexas, baseadas em questões ou problemas desafiadores, que envolvem alunos em design, resolução de problemas, tomada de decisão ou atividades investigativas; dão aos alu-

Tabela 3 – Características da abordagem humanista

Aspecto	**Característica Humanista**
Conhecimento	O conhecimento é construído no decorrer do processo de vir-a-ser de pessoa humana (experiência pessoal e subjetiva). - Sujeito: papel central e primordial na elaboração e criação do conhecimento. Ao experienciar, o homem conhece. - Não é acabado, possui uma característica dinâmica e é inerente à atividade humana. "O único homem que se educa é aquele que aprendeu a aprender" (Rogers)
Educação	- Centrada na pessoa, no aluno. Educação democrática. Finalidade: criar condições que facilitem a aprendizagem, - Objetivo básico: liberar no aluno a capacidade de auto aprendizagem (desenvolvimento intelectual e emocional). - Tornar os alunos pessoas de iniciativa, de responsabilidade, de autodeterminação, com espírito livre e criativo. -Ação pedagógica crítica e consciente
Ensino-Aprendizagem	- Ensino centrado na pessoa (primado do sujeito). - Método não diretivo: dirigir a pessoa à sua própria experiência, para que ela possa estruturar-se e agir. -Personalidades únicas respondendo às circunstâncias únicas. - A aprendizagem deve influir significativamente sobre o comportamento (autodescoberta, autodirigida
Professor-aluno	- O professor é uma personalidade única que assume a função de facilitador da aprendizagem. - O relacionamento entre professor e aluno é sempre pessoal e único. Deve haver uma compreensão empática e o apreço (aceitação e confiança). - O aluno deve responsabilizar-se pelos objetivos referentes à aprendizagem. (Um ser que se autodesenvolve).
Metodologia	- As estratégias instrucionais são secundárias. - Deve-se estimular a curiosidade e o interesse dos alunos. - Os conteúdos devem ser significativos, pesquisados pelos alunos, que devem ser capazes de analisar criticamente os mesmos

Fonte: Adaptado de (MIZUKAMI, 1986)

> nos a oportunidade de trabalhar de forma relativamente autônoma durante longos períodos de tempo; e culminam em produtos realistas ou apresentações (THOMAS, 2000).

Esta pesquisa leva em consideração o conhecimento prévio dos alunos para melhorar o processo de aprendizado. Com isso em mente, será feito o uso das principais características da abordagem PBL, visando capacitar os alunos para realizar pesquisas, integrar teoria e prática e aplicar conhecimento e habilidades para desenvolver uma solução viável para um problema definido (SAVERY, 2015).

Para a ES, o uso do PBL pode contribuir para a melhor compreensão de conteúdos e temas, já que de acordo com (DUCH; GROH; ALLEN, 2001) esse método inclui a capacidade de pensar criticamente, analisar e resolver problemas complexos reais para encontrar, avaliar e usar recursos de aprendizagem apropriados; trabalhar cooperativamente, para demonstrar as habilidades de comunicação eficaz e usar conhecimento e habilidades intelectuais para se tornar um aprendiz contínuo.

2.6 ABORDAGEM PBL

A Aprendizagem Baseada em Problemas (ABP) ou (*Problem-Based Learning-PBL*) , como modelo geral, foi desenvolvida na educação médica em meados da década de 1950 e desde então foi aperfeiçoada, implementada e continua a ganhar aceitação em várias disciplinas (SAVERY, 2015).

PBL muda o paradigma educacional para o professor, pois seu papel muda de apresentador de informações para facilitador do aprendizado num processo de resolução de problemas. Com isso, o papel do aluno também muda, pois agora eles são convidados a conduzir seu próprio aprendizado, já que o professor vai começar a guiá-los em discussões sobre os conteúdos pertinentes e irá intervir quando necessário, fazendo perguntas que investiguem precisão, relevância e profundidade das informações; levantando novas questões para consideração; e promovendo a participação plena e uniforme (MAYO; DONNELLY; SCHWARTZ, 1995).

Em vez de aulas puramente teóricas, onde apenas o professor traz informações sobre determinado assunto, em PBL o professor deve encontrar problemas com base em objetivos de aprendizagem claros, e através destes problemas, irá guiar os alunos a aprender os principais conceitos relacionados ao conteúdo da disciplina. Esses problemas devem ser construídos, não apenas para apresentar aos alunos questões que são importantes para eles, mas também para promover o desenvolvimento de estruturas conceituais (HUNG et al., 2008). O professor também pode apresentar desafios cognitivos onde informações necessárias não são fornecidas, motivando assim que os alunos procurem explicações para o que foi apresentado.

Para a implementação de PBL ser bem sucedida, é de extrema importância a participação do professor na aprendizagem ativa e na construção do conhecimento dos alunos (AMADOR; MILES; PETERS, 2007; DUCH; GROH; ALLEN, 2001). O professor pode planejar intervalos de discussão ou estudos independentes para ajudar os alunos a se aprofundarem em determinados tópicos e assim encontrar recursos úteis para a solução do problema. Ele pode também usar alunos como facilitadores do ensino para ampliar seu alcance instrucional, ministrando monitorias sobre determinados assuntos.

É importante ressaltar que PBL pode suportar o desenvolvimento de uma gama de habilidades: pesquisa, negociação e trabalho em equipe, leitura, escrita e comunicação oral. Para a ES, essa metodologia pode ser usada para melhorar a eficácia da aprendizagem, promovendo a capacidade dos alunos de trabalhar em equipe para resolver problemas e também incentiva o desenvolvimento da aprendizagem autodirigida, cooperação, ética e respeito pelos pontos de vista de outras pessoas (SAVERY; DUFFY, 1995). A seguir são apresentadas as características essenciais da PBL (SAVERY, 2015) que serão utilizadas no *framework* a partir do desenvolvimento de um projeto de software:

- Os estudantes devem ter a responsabilidade pelo seu próprio aprendizado;

- As simulações de problemas utilizados na PBL devem ser mal estruturadas e permitir a livre investigação;
- Colaboração é essencial;
- O que os(as) alunos(as) aprendem durante a aprendizagem auto-dirigida deve ser aplicado de volta ao problema com reanálise e resolução;
- Uma análise final e uma discussão sobre quais conceitos e princípios foram aprendidos são essenciais;
- As atividades realizadas na PBL devem ser aquelas valorizadas no mundo real.

Em (OLIVEIRA; RODRIGUES; GARCIA, 2012) são apresentadas vantagens e benefícios de se aplicar PBL na Ciência da Computação e o tópico mais relevante mostrado foi o de melhoria no desenvolvimento de habilidades, como por exemplo, facilidade na na identificação e resolução de problemas, melhor visão crítica sobre diversos assuntos, trabalho em grupo e aprendizagem autodirigida. Além deste, também foi evidenciado que PBL contribui para a melhor preparação para o mercado de trabalho e melhor gerenciamento do tempo de estudo.

Diversos autores (ALEXANDRE; SANTOS, 2018; SØRENSEN; FALCH; SKOUBY, 2018; SANTOS et al., 2009) também apresentam o uso do PBL para a disciplina de Engenharia de Software, o que ressalta a importância de ser ter um método bem planejado para o ensino desta disciplina, e no qual o PBL agrega bastante valor, devido a sua prática. Assim foram apresentadas diversas maneiras de como ministrar o conteúdo, sempre focado na melhoria do ensino.

2.7 KAIZEN - MELHORIA CONTÍNUA

(IMAI, 1994) descreve esta filosofia da seguinte forma:

> "A essência do kaizen é simples e direta: kaizen significa melhoramento. Mais ainda, kaizen significa contínuo melhoramento, envolvendo todos, inclusive gerentes e operários. A filosofia do kaizen afirma que o nosso modo de vida – seja no trabalho, na sociedade ou em casa – merece ser constantemente melhorado".

A filosofia envolve a definição de padrões e melhorá-los continuamente. O Kaizen é baseado na crença de que tudo pode ser melhorado e nada é status quo. O Kaizen envolve identificar problemas e oportunidades, criar soluções e implementá-las - e, em seguida, passar novamente pelo processo para outras questões ou problemas que foram abordados de forma inadequada.

O Kaizen é baseado em um sistema simples de resolução dos problemas e qualquer ideia, por mais simples que seja, deve ter total atenção. O erro de muitas estratégias de qualidade é de se concentrarem em sistemas muito formais no combate ao desperdício. Na verdade, a metodologia Kaizen, não está ligada à ideia de sofisticação, mas a razões simples e baratas, unidas ao bom senso. A Figura 2 apresenta os 5 princípios fundamentais do Kaizen, que são: Conheça seu cliente, deixe fluir, vá para "Gemba"(refere-se ao local onde o valor é criado; pode ser qualquer "local"onde o provedor de serviços interaja diretamente com o cliente), capacite as pessoas e seja transparente. A implementação desses 5 princípios em qualquer organização é fundamentalmente importante para uma cultura de melhoria contínua bem-sucedida.

Figura 2 – Princípios fundamentais do Kaizen

Fonte: (IMAI, 1994).

Em todas as organizações é crescente a necessidade de resolver problemas e tomar decisões, e isso não é diferente nas IES. Para aplicar o Kaizen neste trabalho, utilizaremos o ciclo PDCA. O Ciclo PDCA ou Ciclo de Deming-Shewhart, é utilizado para melhorar e manter os padrões de desempenho, sendo composto por quatro etapas (MOEN; NORMAN, 2006):

1. Plan: é a etapa na qual o que será feito é planejado, ou seja, nessa etapa o cenário ou problema é analisado e, diante disso, deve ser construído um plano contendo os passos que se pretende realizar.

2. Do: é a etapa prática em que o plano de ação desenvolvido na primeira etapa será executado.

3. Check: é a etapa onde ocorre a coleta de dados e comparação do resultado alcançado com a meta planejada

4. Act: é nessa etapa que são feitas as correções dos erros e falhas no processo, podendo levar ao reinício do ciclo ou a padronização do plano de ação.

A execução do ciclo PDCA é imprescindível para identificar problemas, analisá-los, propor e executar novos procedimentos para que ocorra um processo de melhoria contínua. Dessa forma, no contexto do ensino superior, a melhoria contínua constitui a base do processo de ensino e aprendizagem, sendo esse um processo contínuo de apropriação e geração de novos conhecimentos individuais, que acontece em uma dinâmica de reflexão e ação sobre situações-problema, voltado para o desenvolvimento de competências (VITORINO, 2019).

Em um ambiente em que se busca a melhor forma de passar o conhecimento, a característica de melhoria contínua e a abordagem orientada a processos do PDCA pode trazer resultados significativos, pois cada parte da disciplina passará pelo mesmo estágio várias vezes, garantindo que os erros possam ser corrigidos e adaptados às necessidades da disciplina e à situação da instituição.

O uso de metodologias centradas no aluno, como o PBL, aliados ao PDCA podem despertar o interesse nos alunos de aprimorar suas habilidades em determinados assuntos, visto que esses instigam os alunos a verificar onde estão suas dificuldades, procurar meios de resolvê-las e assim criar seu próprio aprendizado a partir dessa experiência.

O planejamento é o princípio da base educacional e um pré-requisito para que o processo de ensino-aprendizagem seja feito com qualidade. Em sua maioria, os planos de ensino são desenvolvidos tomando como base as experiências do professor, condições de ensino, nível de conhecimento e ambiente de trabalho e pesquisa.

Para a montagem desse plano, as atividades docentes são divididas em várias atividades que por sua vez terão diferentes níveis que serão planejados de formas diferentes. Durante este planejamento, as experiências anteriores podem ser usadas para analisar problemas e definir soluções. A partir disso, o professor pode utilizar de seu conhecimento para melhorar seu plano, afim de atingir a qualidade desejada.

As atividades de execução e verificação também são necessárias para garantir e melhorar a qualidade do ensino. Estas atividades incluem tanto a metodologia dos professores no processo de ensino como os métodos de aprendizagem dos alunos. Portanto, as escolhas apropriadas para estas atividades estão diretamente relacionadas à qualidade do ensino e por isso há vários fatores que o professor deve considerar ao selecionar os métodos para executar e verificar de acordo com o planejamento, como: os objetivos para a disciplina, carga horária, metodologia de ensino, etc.

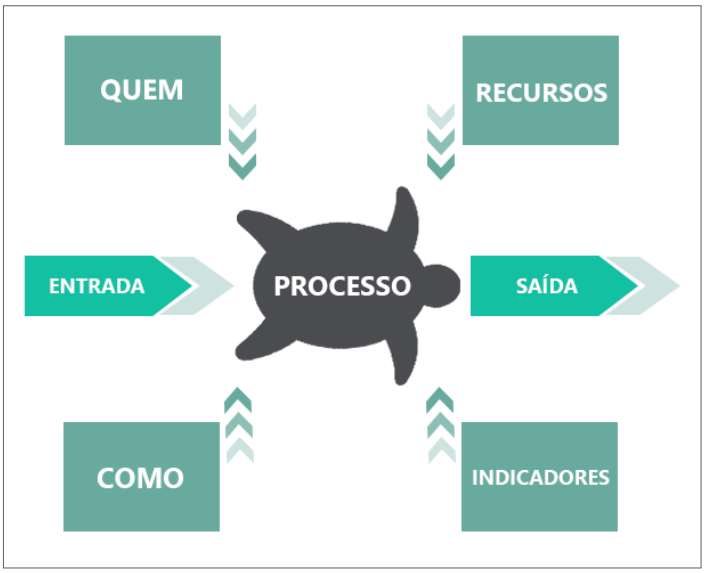
As atividades de ação são imprescindíveis na melhoria contínua da qualidade do ensino. A ação inclui basicamente duas etapas: resumir as lições aprendidas com as experiências, sejam elas bem-sucedidas ou não, e desenvolver padrões adequados para promover essas lições. Com as experiências mal-sucedidas, o professor pode aplicar soluções no próximo ciclo do PDCA, através da segunda etapa da atividade, fazendo com que seja afirmada a

sentença de que a Ação é uma parte essencial para melhorar continuamente a qualidade do ensino.

2.8 DIAGRAMA DE TARTARUGA

Para auxiliar a estruturar e organizar o *framework* proposto, foi utilizado o diagrama de tartaruga, como pode ser visto na Figura 3.

Figura 3 – Diagrama de tartaruga



Fonte: (RAMOS, 2017).

Este diagrama é uma técnica para se registrar um processo de maneira compacta, a fim de tornar possível sua compreensão e posterior melhoria (BARNES, 1977). Esse mapeamento de processos pode ser considerado como uma ferramenta gerencial analítica e de comunicação (HUNT, 1996) que têm a intenção de ajudar a melhorar ou implantar um nova estrutura voltada para o processo de ensino-aprendizagem. O Diagrama de Tartaruga foi adaptado para atender as necessidades do processo educacional e foi dividido em 5 campos:

- Processos: Esse campo visa identificar qual é o processo, formalizando quem é o responsável pela execução e pré estabelecendo qual o resultado esperado
- Entradas: Aqui serão listados os insumos que serão utilizados no processo e que, geralmente, são resultados de processos anteriores.
- Saídas: A saída é o que se origina do processo, aquilo que o processo emite quando termina.

- Como: No campo “como” serão descritos os procedimentos, normas e documentos que nortearão a realização do processo, quais devem ser as práticas adotadas na realização da tarefa
- Quem: Também é preciso listar a equipe de trabalho que realizará a saída, todos que forem necessários à sua boa realização.

2.9 CONCLUSÕES DO CAPÍTULO

Esse capítulo apresentou a fundamentação teórica da tese através da exposição das principais definições e conceitos aqui abordados. Esse trabalho também optou por adotar a classificação da (CURRICULA; SOCIETY, 2013), que define a ES como uma área de conhecimento organizada em um conjunto de unidades de conhecimento que, por sua vez, são constituídas por tópicos e aprendizagens esperadas.

Adicionalmente, foi feita uma análise das diretrizes curriculares mais relevantes com foco em ES como também uma caracterização pedagógica do processo de ensino-aprendizagem. A partir desse cenário, surge a preocupação com o ensino da ES e como ela pode ser aplicada em diferentes contextos dentro dos cursos de graduação em computação. Seu ensino exige um grau muito grande de abstração e conceitos teóricos que, muitas vezes, torna o ensino dos seus conteúdos um verdadeiro desafio para quem leciona e para quem aprende.

Há um consenso de que o ensino da engenharia de software, tradicional e focado em metodologias, deve ser transformado para refletir a demanda por software mais complexo (BAKER; NAVARRO; HOEK, 2005). Os métodos de ensino podem ser ajustados para casos particulares e várias áreas. No caso da engenharia de software, se precisa de elementos complementares que envolvem princípios(conceitos que fundamentam a área), prática(técnicas de resolução de problemas), aplicações(áreas de conhecimento em que os princípios e práticas encontram a sua melhor expressão) e matemática(a base formal que torna possível compreender todo o resto). Para identificar a importância das diversas áreas da ES na visão de diferentes públicos-alvos, o capítulo a seguir pretende fazer uma descrição do cenário brasileiro de ensino de ES.

3 METODOLOGIA DE PESQUISA

Este capítulo descreve como a pesquisa foi conduzida em relação às escolhas dos métodos científicos. A Seção 3.1 mostra o plano de pesquisa estabelecido, a Seção 3.2 apresenta as etapas da pesquisa proposta neste trabalho. A Seção 3.3 apresenta o planejamento e aplicação de um *survey* que foi construído com a intenção de capturar evidências disponíveis sobre o ensino de ES. Na Seção 3.4 é apresentado o conceito da revisão narrativa da literatura, e por fim, a Seção 3.6 traz as conclusões do capítulo.

3.1 PLANEJAMENTO DA PESQUISA

O desenho de pesquisa desta dissertação é baseada em uma abordagem multi-métodos que combina dois ou mais métodos quantitativos e/ou qualitativos em um único estudo (HESSE-BIBER, 2010).

Foi feita uma pesquisa descritiva, já que por meio de técnicas padronizadas de coleta de dados, foram extraídos conhecimentos sobre o ensino da engenharia de software no Brasil. Como definido por (GIL, 2010), o uso de questionário é uma técnica de investigação composta por um conjunto de questões que são submetidas a pessoas com o propósito de obter informações sobre conhecimentos, crenças, sentimentos, valores, interesses, expectativas, aspirações, temores, comportamento presente ou passado, etc. Construir um questionário consiste basicamente em traduzir objetivos da pesquisa em questões específicas. As respostas a essas questões é que irão proporcionar os dados requeridos para descrever as características da população pesquisada.

Também foi feita uma pesquisa bibliográfica, onde utilizou-se de dados extraídos de artigos em periódicos científicos e anais de congresso, permitindo a autora cobrir uma gama de fenômenos muito mais ampla do que aquela que poderia pesquisar diretamente. A pesquisa bibliográfica busca conhecer e analisar as contribuições culturais ou científicas do passado sobre determinado assunto, tema ou problema (GIL, 2010).

As Seções 3.3 e 3.4 mostram como a metodologia de pesquisa foi aplicada nesta dissertação.

3.2 ETAPAS DA PESQUISA

A pesquisa foi conduzida por um processo que mescla as etapas sugeridas por (MARCH; SMITH, 1995), (TAKEDA; VEERKAMP; YOSHIKAWA, 1990), (KASANEN; LUKKA; SIITONEN, 1993) e (HEVNER et al., 2004), sendo elas: consciência do problema, sugestão, desenvolvimento, aplicação e avaliação do *framework*. A partir dessa visão, foi definida uma estratégia de construção do SELF, conforme apresentada na Figura 4.

Figura 4 – Etapas da Pesquisa

Fonte: Elaborado pela autora (2019).

Estas etapas seguem o método de pesquisa conhecido como *Design Science Research* (DSR). Neste método, o conhecimento e a compreensão de um domínio do problema e sua solução são alcançados graças à construção e aplicação de um artefato projetado A construção é o processo de confecção de artefatos para um propósito específico, enquanto a avaliação é a verificação do desempenho dos artefatos como solução desejada (MARCH; SMITH, 1995).

Segundo (HEVNER et al., 2004), numa abordagem pragmática, a DSR não anseia alcançar verdades últimas, grandes teorias ou leis gerais, mas procura identificar e compreender os problemas do mundo real e propor soluções apropriadas, úteis, fazendo avançar o conhecimento teórico da área. Nas subseções a seguir serão apresentados os objetivos e resultados gerais de cada etapa.

3.2.1 Etapa 1: Consciência do Problema

Esta etapa contempla a identificação do problema e o estudo do ambiente/contexto do mesmo. O problema deste estudo tem caráter genérico, pois abrange todos os tipos de IES, porém foi estudado em um contexto específico: uma IES, mais especificamente, a UFPE. Com isso, a solução desenvolvida também é de caráter genérico, tendo sido aplicada em um contexto específico. Neste contexto, percebeu-se uma dúvida comum: uma vez que já foi entendido que o ensino de ES precisa ser adaptado para diferentes contextos, como o professor irá construir uma disciplina de ES que trabalhe seus aspectos teóricos e práticos dentro da limitação de tempo da IES? Evidências (PRIKLADNICKI et al., 2009; GARG; VARMA, 2009; GIMENES, 2015) destacam que os principais problemas no processo de ensino-aprendizagem são:

1. Os discentes têm pouco interesse pelas aulas maciçamente teóricas.

2. Índices de aprendizagem efetiva bastante questionáveis.

3. É dada pouca ênfase ao trabalho em grupo.
4. Pouco tempo para muito conteúdo.
5. Há pouca ou nenhuma interação entre professores e alunos fora da sala de aula.

Esta etapa permitiu a pesquisadora entender que a efetividade do ensino de ES depende das competências que são trabalhadas nos alunos e que essas competências necessárias, muitas vezes, não estão sendo adequadamente abordadas nos cursos de graduação.

Portanto, a pesquisa considera que se faz necessária uma revisão de como os professores podem desenvolver uma disciplina de ES que trate dos conteúdos e competências essenciais para esta área. Não há um consenso em como ensinar ES, pois cada universidade adota seus próprios métodos ou abordagens de acordo com o contexto em que estão inseridas, mas é imprescindível que o professor tenha uma estrutura hierárquica para a construção da sua disciplina.

3.2.2 Etapa 2: Sugestão

Segundo (TAKEDA; VEERKAMP; YOSHIKAWA, 1990), esta é a etapa criativa, que envolve inovação a partir do conhecimento sobre o problema. Visa propor princípios que auxiliem na construção da solução. Para tanto, foram utilizadas pesquisas em periódicos, anais de congressos, dissertações, realização de um *survey* com professores, alunos e profissionais de ES e uma revisão narrativa da literatura para buscar soluções existentes para o problema.

Na avaliação das soluções existentes para a melhoria do ensino de ES através do uso de *frameworks*, constatou-se que há uma deficiência de soluções que instruam o professor a construir uma disciplina de ES com qualidade. A maioria das pesquisas que tratam da melhoria do ensino, focam em abordagens que podem ser usadas em sala de aula, mas não instrui o professor em como ele pode planejar de forma eficiente o uso dessa abordagem.

A Seção 4.4 destaca estas soluções e critica a omissão de sua relação com o processo que vem antes do ensino em si, que é o ato de planejar a disciplina, como também dos processos que se seguem, como avaliar se o aprendizado foi efetivo.

Com o resultado do *survey*, identificou-se a importância das diversas áreas da ES na visão de cada um dos públicos-alvos; qual a ênfase educacional dada para estas áreas nos cursos de graduação em informática e quais áreas e competências os profissionais de Engenharia de Software consideram mais importantes para a profissão, levando em consideração a demanda do mercado.

Portanto, pode-se afirmar que há uma brecha nas pesquisas associadas à melhoria do ensino de ES por não retratar de forma coordenada, em prol do desempenho do professor em relação ao processo de ensino, de como o mesmo pode desenvolver sua disciplina de forma a alcançar a qualidade esperada e quais entradas e saídas associadas ao processo de ensino-aprendizagem.

3.2.3 Etapa 3: Desenvolvimento do *framework*

A etapa de desenvolvimento gerencia os métodos práticos que produzem o artefato. Tendo realizado as etapas de consciência do problema e sugestão, culminando no entendimento do problema e na identificação de conceitos e princípios importantes para a construção do *framework*, foi definida a sua lógica. Em seguida, o *framework* foi desenvolvido de forma gráfica e com descrições textuais.

Neste caso, foi desenvolvido um *framework* conceitual, que significa que o artefato gerado a partir do uso do *framework* não é um software executável , mas sim, um esquema conceitual de dados que posteriormente deverá ser traduzido para um esquema de dados específico do domínio no qual será desenvolvida a aplicação (FILHO, 2000). No caso desta dissertação, o *framework* está relacionado ao apoio do ensino de engenharia de software.

Para auxiliar a estruturar e organizar os processos do *framework* foi utilizado o Diagrama da Tartaruga e o ciclo PDCA. O Diagrama de Tartaruga permite entender todo o funcionamento do processo de construção de uma disciplina, incluindo entradas, saídas, recursos, formas de acompanhamento e demais informações.

O ensino e a avaliação da aprendizagem dos alunos são tarefas inerentemente complexas. Se o objetivo é efetivamente ensinar habilidades, conhecimentos e, em seguida, avaliar o aprendizado do aluno apropriadamente, nenhum roteiro definitivo existe. No entanto, o ciclo PDCA fornece um método sistemático para progredir incrementalmente em direção à meta que se deseja alcançar na disciplina (MERGEN et al., 2013).

Torna-se, portanto uma excelente ferramenta a ser utilizada pelos professores para uma gestão eficiente da sua disciplina. Isso se deve pelo fato de que seu principal enfoque consiste em auxiliar na concretização dos objetivos educacionais, guiando o professor na proposta de planos e projetos que se adaptem a realidade e promovem a melhoria contínua dos mais variados contextos em que uma IES está inserida.

Ao persistir na descoberta e solução de erros pela constante qualificação e aperfeiçoamento do processo de ensino-aprendizagem, demonstra ser uma ferramenta que contribui para a busca do melhor desempenho educacional, fazendo com que seja possível propor melhores resultados e fortalecer a cultura de construção e reconstrução de saberes, em uma área em que as mudanças são constantes.

3.2.4 Etapa 4: Aplicação do *framework*

Esta etapa coloca o *framework* construído à prova, verificando como ele se comporta numa situação real. Esta etapa é fundamental para verificar a viabilidade do *framework*, bem como suas funcionalidades e problemas que possam ocorrer. A partir dos resultados da aplicação, o *framework* poderá ser modificado com o intuito de aperfeiçoá-lo.

A aplicação do *framework* desenvolvido ocorreu na disciplina de Engenharia de Software do curso de Sistemas de Informação da UFPE, durante o semestre de 2019.1. A

aplicação foi coordenada pela pesquisadora e pelo professor da disciplina e será detalhada na Seção 6.1 como o mesmo ajudou o professor a desenvolver a disciplina de forma mais objetiva como também a descobrir problemas inerentes ao processo de ensino-aprendizagem de forma mais precisa.

3.2.5 Etapa 5: Avaliação do *framework*

Avaliação é um componente crucial do processo de pesquisa. É vista como o processo de comparar objetivos predefinidos a resultados observados na prática da atividade de demonstração. O resultado da etapa de avaliação também pode produzir novos problemas que podem levar a novos ciclos de consciência do problema (TAKEDA; VEERKAMP; YOSHIKAWA, 1990).

A avaliação do *framework* foi feita pelo método painel de especialistas. Participaram dessa etapa 3 (três) especialistas na área de ensino de ES com experiência em docência e pesquisa. A avaliação ocorreu de forma individual, onde os especialistas leram o modelo e responderam um questionário estruturado.

De forma geral, os especialistas indicaram que o *framework* oferece um nível de detalhes suficiente para fornecer uma base adequada para seu uso no ensino de ES e que o mesmo é de fácil utilização. Mais detalhes desta avaliação serão apresentados na Seção 6.2.

3.3 *SURVEY*

Um *survey* não é apenas um instrumento para a coleta de informações, ele é um método de pesquisa abrangente para a coleta de informações que visam descrever, comparar ou explicar conhecimentos, atitudes e comportamentos (KITCHENHAM; PFLEEGER, 2008). A finalidade de um *survey* é produzir estatísticas, ou seja, descrições quantitativas ou numéricas de alguns aspectos da população de estudo.

A presente metodologia é caracterizada como descritiva, já que por meio de técnicas padronizadas de coleta de dados, foram extraídas informações sobre as experiências e percepções dos profissionais, alunos e profissionais da área de engenharia de software, representados por uma amostra, em relação aos tópicos da área tidos como relevantes. Como definido por (GIL, 2010), o uso de questionário é uma técnica de investigação composta por um conjunto de questões que são submetidas a pessoas com o propósito de obter informações sobre conhecimentos, crenças, sentimentos, valores, interesses, expectativas, aspirações, temores, comportamento presente ou passado, etc. Construir um questionário consiste basicamente em traduzir objetivos da pesquisa em questões específicas. As respostas a essas questões é que irão proporcionar os dados requeridos para descrever as características da população pesquisada.

Por fim, a abordagem adotada na análise dos dados é de caráter quantitativa e qualitativa. Quantitativa porque o questionário disponibilizado apresenta questões objetivas,

e nesta abordagem foram usadas análises estatísticas nos dados coletados. A abordagem qualitativa foi usada para entender como ocorre o processo de ensino-aprendizagem de ES em diferentes partes do Brasil.

3.3.1 Público-alvo

Como público-alvo do *survey*, definiu-se:

- Professores de disciplinas de Engenharia de Software;
- Alunos graduados em cursos de informática com até dois anos de formação;
- Profissionais que exerçam papel de gestor de times de engenharia de software em empresas de TI.

  Foram excluídos os participantes que não atendiam estes critérios ou que não estavam motivados a participar da pesquisa. Estes critérios de inclusão e exclusão foram divulgados no protocolo do *survey*.

3.3.2 Questão de Pesquisa

O presente *survey* pretende responder a seguinte questão de pesquisa:

"Há diferença entre a importância dada às áreas da engenharia de software vistas na graduação com as necessidades da indústria de software?"

Com o intuito de encontrar soluções para o estudo em questão, foram levantadas hipóteses de pesquisas e que surgiram com base na análise do referencial teórico do tema. A partir desta análise, foram formuladas as seguintes hipóteses:

HP1. É possível relacionar o nível de aprendizado dos alunos com a importância dada pelos professores ao tópicos da engenharia de software.

HP2. As competências mais importantes para um profissional de ES estão relacionadas com os conteúdos mais ministrados pelos professores.

3.3.3 Questionários

Foram utilizados como instrumentos para aplicação e coleta de dados do *survey* três questionários auto administrados, com questões contendo um conjunto de respostas pré-definidas. Estas questões abordaram o nível de aprendizagem dos alunos em relação às áreas de conhecimento da engenharia de software, como também a importância dessas áreas na visão dos professores e profissionais. As áreas de conhecimento utilizadas foram identificadas após a revisão dos currículos de referência da ACM/IEEE (BOURQUE; ROBERT,) e o SWEBOK (BOURQUE; FAIRLEY et al., 2014) e foram identificadas 15 área: 1) Requisitos de Software 2) Design de Software 3)Construção de Software 4)Teste de Software 5)Manutenção de Software 6)Gerenciamento de Configuração 7)Gerenciamento de

Engenharia de Software 8)Processos de Engenharia de Software 9)Modelos e Métodos de Engenharia de Software 10)Qualidade de Software 11)Prática Profissional de Engenharia de Software 12)Economia de Engenharia de Software 13)Fundamentos de Computação 14)Fundamentos de Matemática 15)Fundamentos de Engenharia.

Para as abordagens do ensino de engenharia de software, identificou-se o trabalho de (PRIKLADNICKI et al., 2009), que destaca os principais métodos de ensino e abordagens de avaliação adotadas nas disciplinas de Engenharia de Software no Brasil. Tomando esta pesquisa como base, foram selecionadas para análise os seguintes elementos de ensino: A) Abordagens de Ensino; e B) Estratégias de Avaliação.

Estes questionários foram disponibilizados na web, utilizando a ferramenta *Google Forms*, que permite a coleta de informações através de uma pesquisa personalizada que é automaticamente ligada a uma planilha.

Além disso, foram incluídas questões demográficas a fim de caracterizar os participantes. Antes da coleta de dados, foi realizado um pré-teste de aplicação do *survey* com 2 professores, 2 profissionais e 4 alunos a fim de identificar possíveis inconsistências e o tempo necessário para responder todas as questões de pesquisa dos questionários. Os participantes responderam aos questionários e indicaram que o tempo para responder todas as questões foi em média 25 minutos para professores e profissionais e de 15 minutos para alunos.

3.4 REVISÃO NARRATIVA DA LITERATURA

A revisão narrativa é constituída por uma análise ampla da literatura, sem estabelecer uma metodologia rigorosa e replicável em nível de reprodução de dados e respostas quantitativas para questões específicas, como explicitam (VOSGERAU; ROMANOWSKI, 2014). No entanto, é fundamental para a aquisição e atualização do conhecimento sobre uma temática específica, evidenciando novas ideias, métodos e subtemas que têm recebido maior ou menor ênfase na literatura selecionada (ELIAS et al., 2012).

Por ser uma análise bibliográfica com o objetivo de capturar informações sobre o uso de *frameworks* como ferramenta para melhorar o ensino de ES, foram recuperados artigos indexados nas bases de dados IEEE, ACM e Google Schoolar durante o mês de outubro de 2018, tendo como período de referência os últimos 15 anos.

Foram empregados os termos de indexação ou descritores *framework, software engineering, education, teaching e learning* de forma combinada. O critério utilizado para inclusão das publicações era ter as expressões utilizadas nas buscas no título ou palavras-chave, ou ter explícito no resumo que o texto se relaciona à associação do uso de *frameworks* para o ensino de ES. Os artigos excluídos não apresentavam o critério de inclusão estabelecido e/ou apresentavam duplicidade, ou seja, publicações recuperadas em mais de uma das bases de dados. Também foram excluídas dissertações e teses.

Após terem sido recuperadas as informações necessárias, foi iniciada a leitura dos títulos e resumos, e ocorrendo exclusão de 12 publicações nessa etapa. Posteriormente, foi realizada a leitura completa de 8 artigos. A partir daí, iniciou-se a análise da fundamentação teórica dos estudos e aplicação dos *frameworks* em sala de aula, bem como a observação das características gerais dos artigos, tais como ano de publicação e língua, seguido de seus objetivos. Por fim, realizou-se a apreciação da metodologia aplicada, resultados obtidos e discussão. Especificamente, para analisar a produção científica identificada, não se utilizaram técnicas qualitativas e/ou quantitativas específicas de tratamento de dados, tendo sido feita a análise de cada um dos textos.

3.5 LIMITAÇÕES DA PESQUISA

Dentre as limitações que podem ter influenciado de alguma forma os resultados da pesquisa, tem-se o fato do painel ter envolvido apenas 3 especialistas, o que faz com que o grau de generalização dos resultados seja muito baixo. No entanto, destaca-se que esses especialistas atuam como professores/pesquisadores na área de ensino de ES em diferentes instituições públicas de ensino (UFRPE, CESAR School e IFPE). Esses professores possuem uma média de 8 anos de experiência docente, sendo que todos já realizaram projetos práticos e discussão de casos práticos no ensino de ES.

A autora considera que, mesmo com uma pequena amostra, não se pode considerar que os resultados da avaliação do SELF, descritos na 6.2, sejam inválidos. Uma quantidade maior de docentes respondentes seria recomendado para pesquisas futuras.

A segunda limitação considerada nesta pesquisa refere-se à participação da autora na função de monitora da disciplina em que o SELF foi aplicado. Neste caso, entende-se que a pesquisa foi realizada com um viés qualitativo e seus resultados foram interpretados sob esta condição. Entende-se que, para aumentar a credibilidade e generalização da pesquisa, é preciso que o SELF seja aplicado em outras disciplinas, sob a condução de outros pesquisadores.

3.6 CONCLUSÕES DO CAPÍTULO

Um dos principais objetivos desta pesquisa é encontrar evidências de que o uso de atividades práticas que são focadas no aluno melhoram o processo de ensino-aprendizagem no sentido de maior aproveitamento do conteúdo ensinado.

Primeiramente realizou-se um *survey* com o objetivo de coletar informações sobre as opiniões de alunos, professores e profissionais representados por uma amostra dessas populações, em relação ao ensino e trabalho em ES.

Após o *survey*, foi feita uma análise narrativa da literatura, para capturar quais as principais abordagens práticas que vêm sendo adotadas no ensino de ES. Posteriormente,

é realizada uma análise destes trabalhos relacionados, destacando suas vantagens e desvantagens em relação ao propósito desta pesquisa.

Buscando atender a esses objetivos, este capítulo apresentou as etapas deste trabalho e os principais métodos de pesquisas utilizados que serviram para fundamentar a definição do *framework* e seus resultados são apresentados no Capítulo 4.

4 RESULTADOS

O ponto de partida de toda pesquisa é definido por uma razão que inicia a investigação. A Seção 4.1 descreve os resultados do survey aplicado. A Seção 4.2 apresenta o resultado de uma revisão narrativa da literatura que foi feita para ser usada como mais uma base para a proposta. Na Seção 4.3 a autora faz uma discussão sobre os resultados da revisão. Por fim, a Seção 4.5 traz as conclusões do capítulo comas justificativas que levaram a autora a elaborar o *framework* que será apresentado no próximo capítulo.

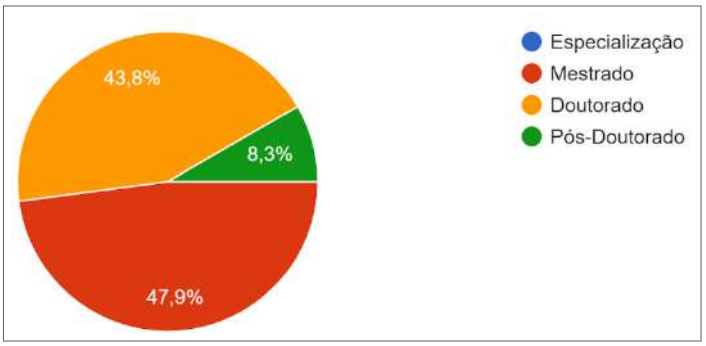
4.1 RESULTADOS DO *SURVEY*

Os dados foram coletados durante o período de 15 de Março a 10 de Maio de 2018. Durante esse período, recebeu-se respostas completas de 98 participantes, sendo 48 professores, 23 profissionais e 27 alunos. Obteve-se respostas de 45 instituições de ensino públicas e privadas e de profissionais que atuam no mercado de TI em 9 estados diferentes. O total de participantes representam 20 estados do Brasil, cerca de 77% do total de estados. Destes, 63% são da região Nordeste, 3% do Norte, 8% do Sul, 11% do Centro-Oeste e 15% do Sudeste.

4.1.1 Professores

Quanto à titulação dos professores,como pode ser visto na Figura 5, 47,9% possuem mestrado; 43,8% possuem doutorado e 88,3% possuem pós-doutorado. Destes, 85,4% já trabalharam na indústria de software.

Figura 5 – Titulação dos professores



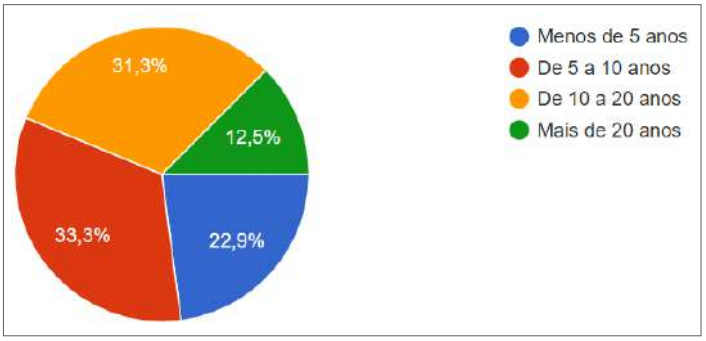
Fonte: Elaborado pela autora (2019).

4.1.1.1 Tempo de profissão e trabalho na indústria

A Figura 6 mostra que dos 48 professores, 12,5% tem mais de 20 anos de tempo de profissão; 31,3% têm de 10 a 20 anos; 33,3% tem de 5 a 10 anos; 22,9% tem menos de 5

anos. Dos professores que já trabalharam na indústria de software 41,5% passaram mais de 5 anos neste ramo de atuação.

Figura 6 – Tempo de profissão



Fonte: Elaborado pela autora (2019).

O tempo passado na indústria impactou positivamente a maneira do professor de planejar e executar a disciplina de Engenharia de Software. Um dos impactos mais citados têm relação com as necessidades práticas da indústria e que muitas vezes não são levadas em consideração no ambiente de sala de aula. A seguir podemos ver algumas respostas obtidas na questão que tratava a relação da sala de aula com o trabalho no ambiente da indústria:

> "Entender os contextos práticos das aplicações teóricas na indústria possibilita levar para a sala de aula exemplos mais significativos do que os que são dados nos livros. O foco na prática da construção de software é outro fruto desse tempo de trabalho na indústria"

> "Trazer aprendizado prático a conceitos que são entediantes do ponto de vista teórico. Atualização de modelos tradicionais para adaptações ou uso de modelos mais adequados a região e contexto da indústria local e regional."

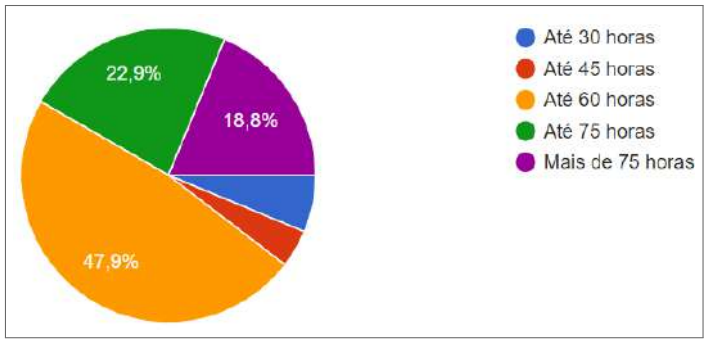
> "Sendo mais pragmático na forma de ensino. A intenção era trazer para sala de aula o conhecimento no nível que realmente se precisa no mercado de trabalho."

4.1.1.2 Carga Horária e plano de ensino

Como pode ser visto na Figura 7, a média semestral da carga horária das disciplinas ministradas é de até 60h para 47,9% dos professores, de até 75h para 22,9% dos professores e mais de 75h para 18,8% dos professores.

Para 58,3% dos respondentes esta carga horária é vista como suficiente para que se trabalhe os conteúdos necessários, e 41,7% acham que essa carga horária é insuficiente. Quando questionados sobre a existência de tópicos que o professor gostaria de incluir no

Figura 7 – Carga Horária



Fonte: Elaborado pela autora (2019).

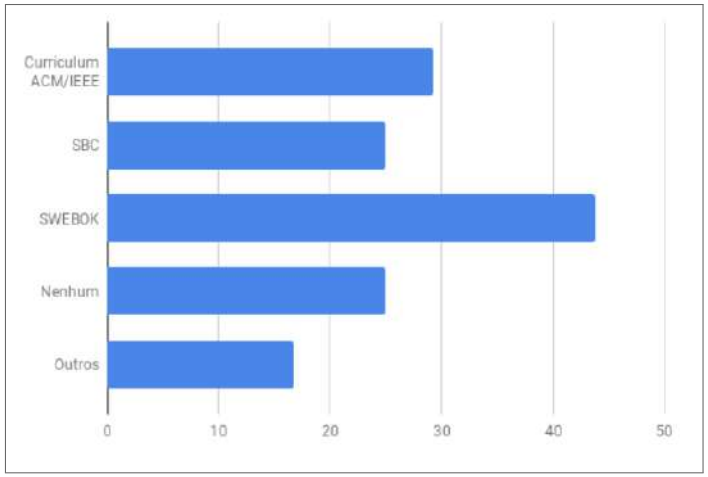
plano de ensino, mas não são incluídas por motivos maiores, o principal motivo citado foi a falta de tempo para trabalhar muito conteúdo.

Alguns dos professores que se mostraram satisfeitos com a abrangência dada aos tópicos da Engenharia de Software, o fizeram porque os tópicos são divididos em mais de uma disciplina, havendo assim tempo suficiente para cobrir tudo que é considerado relevante. Alguns dos tópicos citados que ficam de fora do plano de ensino são: métodos formais, modelos de maturidade, análise de segurança e gerência de configuração.

4.1.1.3 Currículos de Referência adotados na ementa de engenharia de software

Em relação aos currículos de referência adotados pelos professores, na Figura 8 observa-se que 25% utilizam o currículo de referência da SBC, 29,2% utilizam o Curriculum Guidelines da ACM/IEEE, 43,8% utilizam o SWEBOK, 25% indicaram não usar currículos de referência e 16,8% adotam outras referências para criação do plano de ensino, como PMBOK, artigos e livros da área. 15 dos 48 professores respondentes utilizam mais de um currículo.

Figura 8 – Currículos Adotados

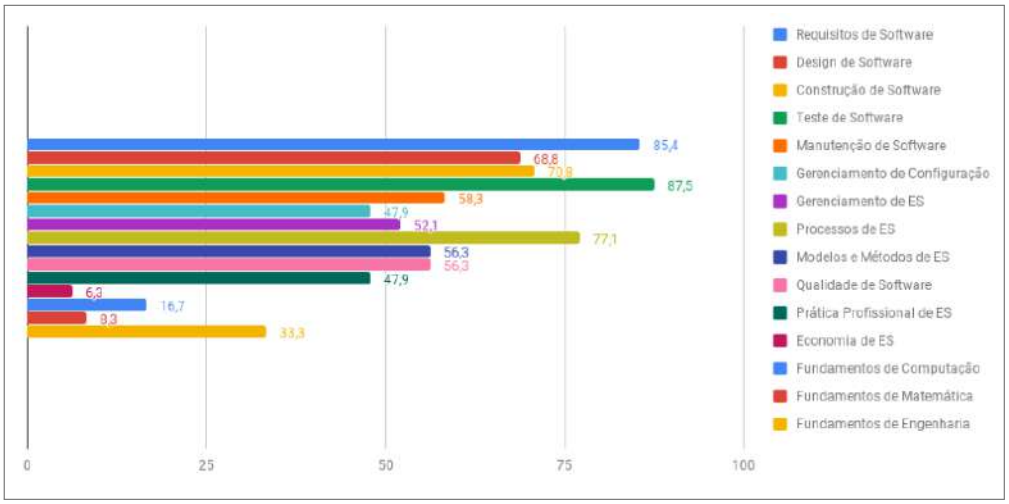


Fonte: Elaborado pela autora (2019).

4.1.1.4 Áreas de conhecimento contempladas na ementa de Engenharia de Software

Há várias formas de destacar a relevância de uma área de conhecimento, mas neste *survey*, a forma usada foi a importância dada pelos professores a determinada área durante as aulas, considerando-se como relevantes as áreas que são adotadas por mais de 70% dos professores. Na Figura 9, observa-se que as áreas de Teste de Software, Requisitos de Software, Processos de Engenharia de Software e Design de Software são consideradas as áreas de maior relevância. Também observa-se que as áreas de Economia de Engenharia de Software,Fundamentos de Matemática, Fundamentos da Computação e Fundamentos de Engenharia são as áreas consideradas de menor relevância.

Figura 9 – Áreas de conhecimento nas ementas de ES



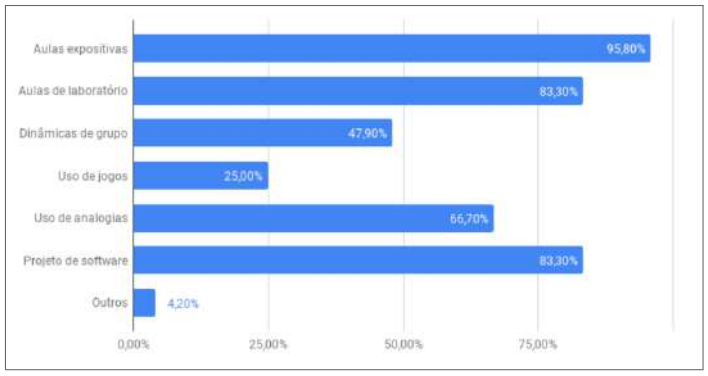
Fonte: Elaborado pela autora (2019).

4.1.1.5 Abordagens de Ensino das disciplinas de engenharia de software

Em relação às abordagens de ensino utilizadas pelos professores, 83,3% dos professores adotam abordagens de ensino que focam nos alunos, como execução de projetos, 66,7% adotam o uso de analogias, 25% usam jogos e 47,9% utilizam dinâmicas de grupo; 95,8% adotam abordagens onde o professor é o principal fornecedor da informação, como aulas expositivas e 83,3% adotam aulas de laboratório. Um professor ainda indicou se utilizar de pesquisas de campo com clientes/stakeholders ou mercado,o que entrou para a categoria "outros". A Figura 10 apresenta os processos de ensino que poderiam ser escolhidos pelos professores e suas porcentagens de escolha.

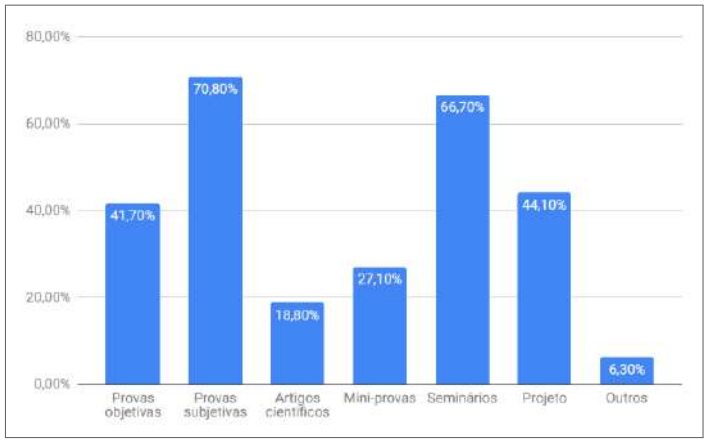
Quanto aos mecanismos de avaliação utilizados ao longo da disciplina, a Figura 11 demonstra que os mais relevantes, levando em consideração a quantidade de adoção, são provas subjetivas, seminários e implementação de projetos.

Figura 10 – Abordagens de ensino



Fonte: Elaborado pela autora (2019).

Figura 11 – Mecanismos de avaliação



Fonte: Elaborado pela autora (2019).

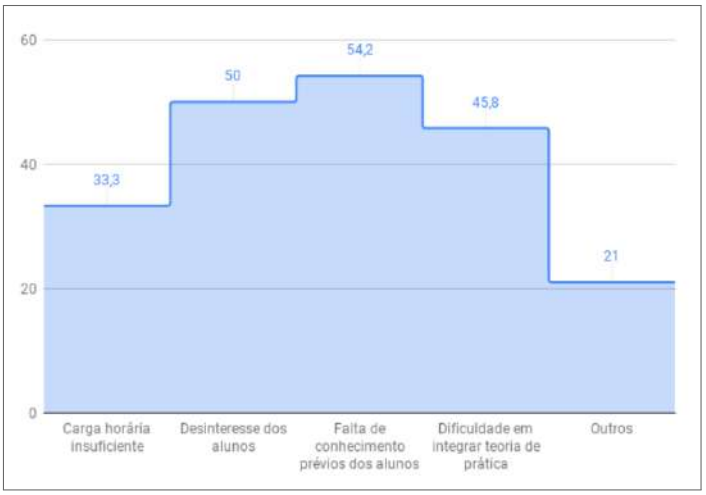
4.1.1.6 Desafios do ensino em Engenharia de Software

Em relação aos principais desafios de ensino enfrentados pelos professores, observa-se na Figura 12 que mais da metade deles (54,2%) informaram que a falta de conhecimento prévio dos alunos, quando há pré-requisitos para a disciplina dificulta o ensino. Dentre os desafios que foram marcados como "Outros", estão: ensinar em pequena escala o que resolve problemas de larga escala; e dificuldade em trazer exemplo reais, pois os alunos estão muito habituados a fazer software simples, que de fato não precisa de práticas de engenharia de software.

4.1.2 Alunos

Para os alunos, 48,1% são do curso de sistemas de informação; 7,4% de engenharia da computação; 37% de ciência da computação e 7,4% de engenharia de software. Dos 27 alunos participantes, 22,2% concluíram a graduação em 2015.2; 14,8% em 2016.1; 11,1% em 2016.2; 22,2% em 2017.1 e 29,6% em 2017.2.

Figura 12 – Desafios do ensino em ES

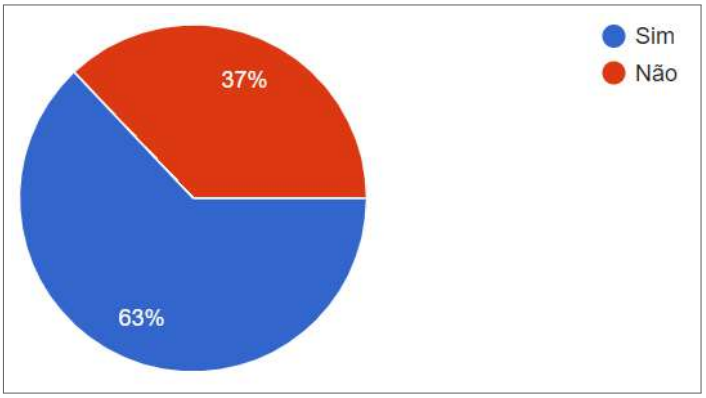


Fonte: Elaborado pela autora (2019).

4.1.2.1 Carga horária e atividades práticas

Como pode ser visto na Figura 13, para 63% dos aluno a carga horária para a área de Engenharia de Software foi suficiente para um bom aprendizado e 96,3% participaram da realização de algum projeto de desenvolvimento ao longo do curso. Na visão de 70,4% dos alunos, o projeto serviu apenas para obter nota na disciplina e apenas 3,7% enxergaram o projeto como uma forma de aprender os conceitos de Engenharia de Software na prática.

Figura 13 – Carga horária na visão dos alunos



Fonte: Elaborado pela autora (2019).

Em relação a atividades práticas que foram passadas pelos professores, e o quanto elas contribuíram para o aprendizado dos alunos, em sua maioria as respostas foram positivas levando em consideração que os alunos puderam aprender a colocar em prática toda a teoria vista em sala de aula. A seguir podemos visualizar algumas das respostas positivas:

> "Foi excelente. Aprendemos na prática como utilizar o que conhecíamos somente na teoria. Ter o conhecimento teórico é importante, porém a prática é

> essencial para que consolide o aprendizado. É durante a prática que podemos realmente entender o funcionamento e utilização dos assuntos discutidos em sala."
>
> "Contribuíram para construir uma base prática fundamentada em conceitos teóricos comprovados, e isso levo para toda minha vida profissional"

No entanto, alguns alunos se demonstraram insatisfeitos com a forma que as atividades foram implementadas pelo professor.

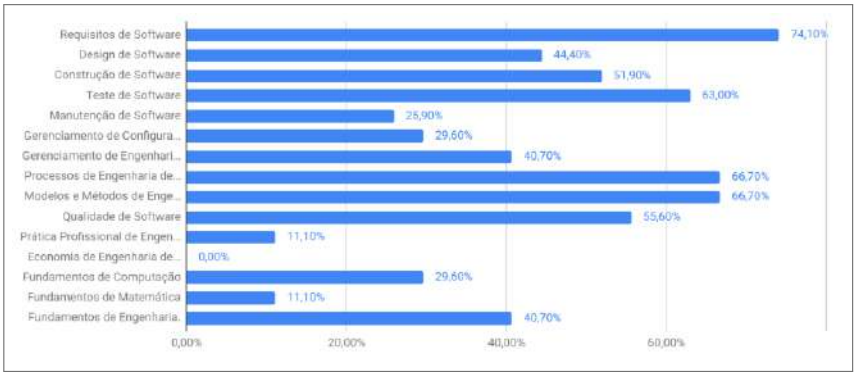
> "Pouco, geralmente atividades descoordenadas que não proporcionavam a execução de um projeto lógico até o final, muita teoria e pouca prática."
>
> "A disciplina de Eng. Soft. especificamente não foi tão boa. Não nos foi ensinado corretamente conceitos importantes como testes automatizados, design patterns, architectural patterns, etc. Nós fingimos aplicar um processo estilo espiral, e fingimos usar git. No fim das contas, não absorvemos tanta coisa. Sabíamos o que existia, mais ou menos, mas não sabíamos usar."

4.1.2.2 Aprendizagem em Engenharia de Software

Observa-se que a área de Requisitos de Software possui a maior porcentagem de aprendizado contando com 74,1% dos alunos assinalando que esta foi a área que mais aprenderam ao longo das disciplinas.

Conforme é apresentado na Figura 14, além da área de Requisitos de Software, as áreas de Processos de Engenharia de Software, Modelos e Métodos de Engenharia de Software e Teste de Software possuem grande aprendizagem, acima de 60%. Já as áreas de Manutenção de Software, Gerenciamento de Configuração, Fundamentos da Computação, Prática Profissional de Engenharia de Software e Fundamentos de Matemática possuem menor aprendizagem, abaixo de 30%.

Figura 14 – Aprendizagem em ES



Fonte: Elaborado pela autora (2019).

Quando questionados sobre o que poderia ser feito para melhorar o processo de aprendizagem, os participantes puderam dar sugestões. Algumas destas sugestões podem ser vistas a seguir:

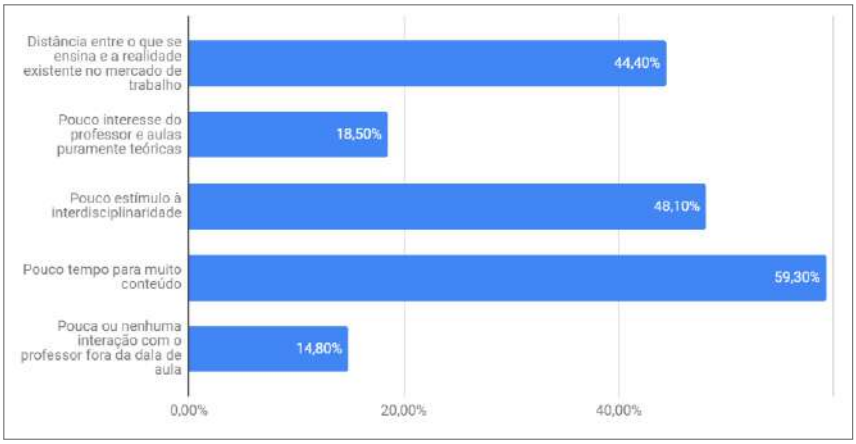
> "Distribuir melhor as disciplinas no curso, incentivar a interação com outros cursos propondo projetos com envolvimento de várias áreas"

> "Gostaria que nas disciplinas de Engenharia de Software os alunos fossem instigados a praticar toda a teoria ensinada, com professores que focassem que questões práticas e se desvinculassem de normas técnicas que são fortemente apresentadas durante o conteúdo nas disciplinas. Seria interessante também, o ensino sobre tipos de arquiteturas e como elas realmente são aplicadas na prática."

4.1.2.3 Desafios do Aprendizado em Engenharia de Software

Em relação aos principais desafios do aprendizado enfrentados pelos alunos, a Figura 15 mostra que mais da metade deles (59,3%) informaram que o pouco tempo para muito conteúdo é o principal problema que influencia negativamente no aprendizado, seguido por pouco estímulo à interdisciplinaridade(48,1%) e a distância entre o que se ensina e a realidade existente no mercado de trabalho (44,4%).

Figura 15 – Desafios do aprendizado em ES



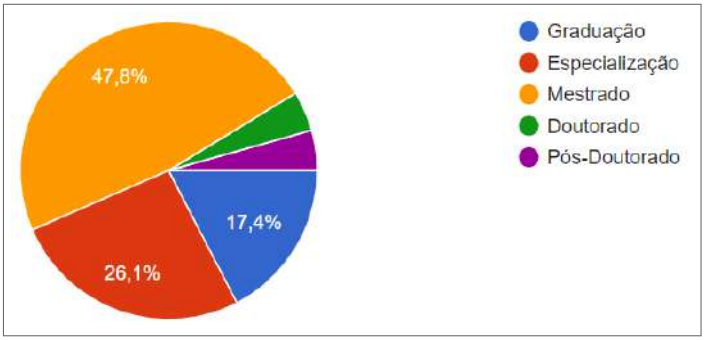
Fonte: Elaborado pela autora (2019).

4.1.3 Profissionais

Para os profissionais, em relação o tempo de atuação na área, 8,7% tem mais de 20 anos; 34,8% têm entre 10 e 20 anos; 34,8% tem de 5 a 10 anos; e 21,7% têm menos de 5 anos. Destes, 39,1% atuam em fábricas de software; 21,7% em empresas de produto;

17,4% em institutos de inovação e 21,5% em outros perfis de empresas como *startup, e-commerce*, governo, etc. Como pode ser visto na Figura 16, a última formação de 47,8% dos profissionais foi o mestrado, 26,1% possuem titulação de espacialista e 17,4% são graduados.

Figura 16 – Titulação dos profissionais

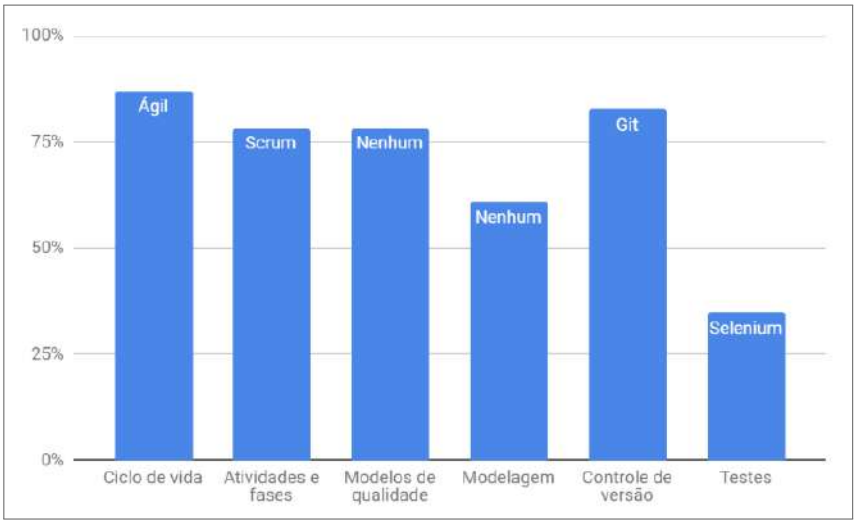


Fonte: Elaborado pela autora (2019).

4.1.3.1 Processos de desenvolvimento

Para definir quais processos de desenvolvimento são mais utilizados na indústria, os participantes foram questionados sobre quais referenciais eram utilizados quanto ao ciclo de vida; atividades e fases; modelos de qualidade; modelagem; controle de versão e testes. A Figura 17 mostra quais referenciais são mais usados para cada etapa.

Figura 17 – Processos de desenvolvimento

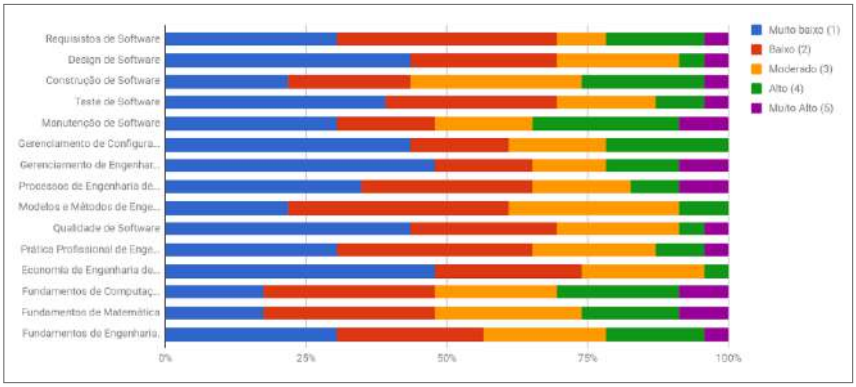


Fonte: Elaborado pela autora (2019).

4.1.3.2 Percepção sobre recém formados

Para poder capturar essa percepção, foi perguntado qual nível de conhecimento das áreas de Engenharia de Software os profissionais observam que os recém-formados possuem assim que chegam na indústria. O nível de conhecimento foi posto em uma escala de 1 a 5, sendo 1 conhecimento considerado muito baixo e 5 considerado conhecimento muito alto. O conhecimento das áreas foi calculado sobre a proporção de respostas dos participantes, sendo demonstrado na Figura 18.

Figura 18 – Percepção sobre recém formados



Fonte: Elaborado pela autora (2019).

Em relação às competências que são vistas como mais importantes para um recém-formado que deseja atuar na área de engenharia de software a de "Identificar, especificar e validar os requisitos, projetar, implementar, verificar, implantar e documentar soluções de software baseadas no conhecimento apropriado de teorias, modelos e técnicas"é tida como a mais importante pelos profissionais (65,2%) e também é a que os recém formados menos possuem de acordo com 56,5% dos respondentes. A segunda competência tida como mais importante por 56,5% dos profissionais é a de "Empregar metodologias que visem garantir critérios de qualidade ao longo de todas as etapas de desenvolvimento de uma solução computacional."

4.1.4 Discussão dos Resultados

4.1.4.1 Sobre a adoção de currículos de referência

Em relação a adoção de currículos de referência, observa-se que 75% dos professores adotam algum currículo de referência para guiar sua ementa nas disciplinas de ES. O uso destes currículos servem como guias para o estabelecimento da engenharia de software como uma disciplina reconhecida de engenharia, além de definir o perfil profissional e acadêmico esperado para os estudantes da área.

Também apresentam uma estruturação e detalhamento das matérias, como carga horária, tópicos a serem abordados e aprendizagens esperadas para cada um destes tópicos.

Sem a referência destes currículos, os professores podem criar ementas que não condizem com as diretrizes nacionais e internacionais de ensino.

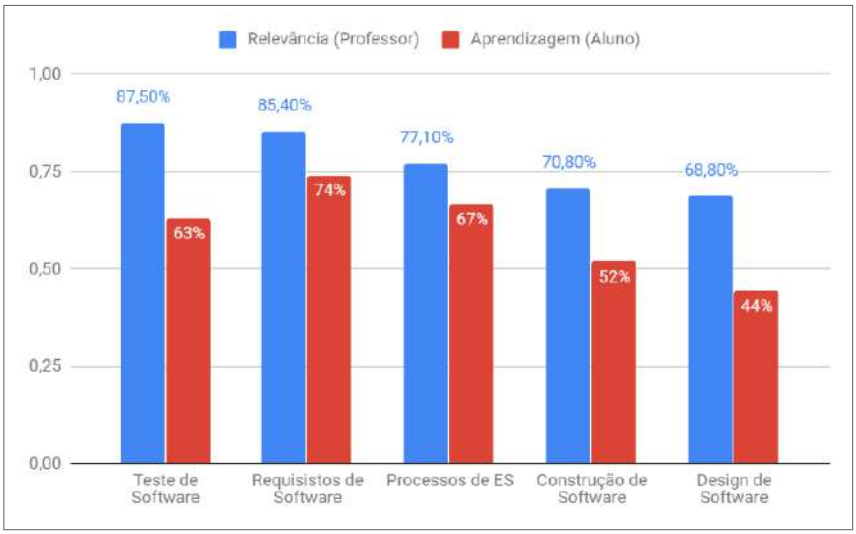
Porém, os currículos devem ser utilizados de forma cautelosa, visto que pode haver uma falta de compatibilidade entre as diretrizes e a realidade dos programas de graduação.

4.1.4.2 Sobre a relevância das áreas de conhecimento

Identificando as 5 áreas de conhecimento mais adotadas pelos professores nas ementas das disciplinas de engenharia de software, é possível correlacionar o percentual de áreas relevantes com o percentual de aprendizagem dos alunos, conforme pode ser visto na Figura 19.

Acredita-se que a relevância da área de Requisitos de Software se deve ao fato de que essa área representa as necessidades reais dos usuários, clientes e outras partes interessadas afetadas por um sistema. Para que essas necessidades sejam atendidas, o profissional de engenharia de software deverá ter conhecimento em elicitação e análise das necessidades das partes interessadas e a criação de uma descrição adequada do comportamento e qualidades do sistema desejado, juntamente com restrições e suposições relevantes. Neste contexto, pode ser inserido área de Teste de Software.

Figura 19 – Relevância das das áreas de conhecimento de ES



Fonte: Elaborado pela autora (2019).

Uma propriedade essencial de um requisito de software é que deve ser possível validar que o produto acabado o satisfaz. Requisitos que não podem ser validados são apenas "desejos". Uma tarefa importante é, portanto, o planejamento de como testar cada requisito. Na maioria dos casos, a concepção de testes de aceitação faz isso para a forma de como os usuários finais tipicamente realizam negócios usando o sistema (BOURQUE; FAIRLEY et al., 2014).

A área de conhecimento do processo de ES está relacionada com a definição, implementação, controle, e proposta de mudança no próprio processo. Esta área envolve vários outros processos, como o de desenvolvimento o de gerenciamento, e o de qualidade. Sua relevância pode ser entendida quando se leva em consideração que os profissionais da área devem ter a capacidade de entender o desenvolvimento de software como um processo a fim de assegurar prazos, custos e a qualidade do produto a ser desenvolvido.

A quarta área de maior relevância, Construção de Software, refere-se à criação detalhada do trabalho de desenvolvimento de software, através da combinação de codificação, verificação e teste, ou seja, não basta definir um processo, é necessário ter habilidades práticas para construir um software que tenha valor para o cliente.

Por fim, a área de Design de software, consiste em duas atividades que se encaixam entre a áreas de Requisitos de Software e a Construção de Software. São elas: desenho arquitetônico de software (às vezes chamado design de alto nível), que desenvolve estrutura e organização de alto nível do software e identifica os vários componentes; e design detalhado do software, que especifica cada componente em detalhes suficientes para facilitar sua construção (BOURQUE; FAIRLEY et al., 2014).

Também é importante destacar que conforme mostrado na subseção 2.2.2, a Diretriz Curricular IS2010 (TOPI et al., 2010) tem o foco dos conteúdos mais em negócio (gerenciamento e planejamento de projetos) do que em computação. Embora ela seja uma base para os cursos de SI, muitos programas de graduação em SI não a incorporam totalmente (HERBERT et al., 2014; WIBISONO; NISAFANI, 2013; HWANG; MA; WANG, 2015), pois esse direcionamento faz com que seus tópicos não atendam as demandas do mercado.

Embora o IS 2010 não o inclua mais como um tópico central, a força-tarefa reconheceu que “um argumento forte pode ser feito para a inclusão de programação, pensamento computacional, dados estruturas e material relacionado em um programa de SI ”(TOPI et al., 2010).

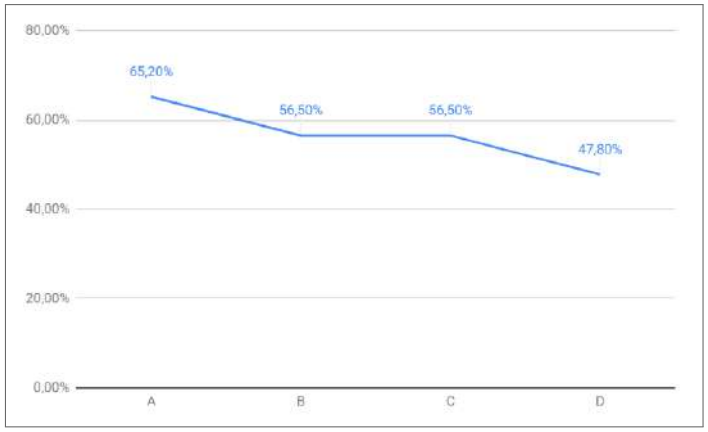
Na visão da autora, estes fatos mostram que a maioria dos programas de SI ainda valorizam a programação como uma habilidade essencial de SI e, portanto, podem estar relutantes ou não querem sacrificar esse curso em favor de conteúdo com foco no gerenciamento. A natureza dinâmica de SI significa que os programas de SI não podem proporcionar um ciclo de vida prolongado, se desejam permanecer relevantes para as necessidades da indústria. Ao contrário dos programas em disciplinas mais estáticas, os programas de SI devem se adaptar às mudanças no mercado, incluindo a integração oportuna de recomendações curriculares contemporâneas.

4.1.4.3 Sobre competências em ES

(FLEURY; FLEURY, 2001) define competência como: um saber agir responsável e reconhecido, que implica mobilizar, integrar, transferir conhecimentos, recursos e habilidades, que agreguem valor econômico à organização e valor social ao indivíduo. A competência

do indivíduo não é um estado, não se reduz a um conhecimento ou “now how” específico. Segundo (BOTERF et al., 1995) competência é um saber agir responsável e que é reconhecido pelos outros. Implica saber como mobilizar, integrar e transferir os conhecimentos, recursos e habilidades, num contexto profissional determinado. Para o contexto de ES foram utilizadas competências que levam em consideração o conhecimento básico necessário para um engenheiro de software, a partir do SWEBOK (BOURQUE; FAIRLEY et al., 2014). A Figura 20 mostra as competências que os profissionais marcaram como mais importantes para atuar na área de ES.

Figura 20 – Competências em ES



Fonte: Elaborado pela autora (2019).

(A) Identificar, especificar e validar os requisitos, projetar, implementar, verificar, implantar e documentar soluções de software baseadas no conhecimento apropriado de teorias, modelos e técnicas.

(B) Manter software, corrigindo falhas, adaptando-o ao seu contexto, identificando e implementando melhorias, migrando softwares legados e retirando software.

(C) Empregar metodologias que visem garantir critérios de qualidade ao longo de todas as etapas de desenvolvimento de uma solução computacional

(D) Construir software para informatização de sistemas avaliando sua qualidade técnica, testando o seu funcionamento, e validando seu atendimento às necessidades de eficácia, eficiência, efetividade e sustentabilidade desses sistemas.

Os profissionais apontaram que as competências mais importantes para a indústria estão atreladas às áreas de conhecimento de Requisitos de Software(A), Manutenção de Software(B), Processos de ES(C), Modelos e métodos de ES(C), Construção de Software(D) e Teste de Software(D). Embora tenhamos de ter muito cuidado ao interpretar os resultados, o mapeamento dessas competências oferece subsídios no planejamento de

uma disciplina de ES e no dimensionamento das atividades práticas necessárias à formação dos alunos.

Analisando os resultados obtidos, observa-se que o cenário atual de ensino está se adequando a indústria quando se leva em consideração que os conteúdos mais ensinados estão relacionados com as competências mais importantes para um engenheiro de software. Porém, essas competências também envolvem habilidades não-técnicas que muitas vezes não são trabalhadas de forma efetiva dentro da sala de aula. Esta constatação salienta a importância de se adotar abordagens de ensino que deixem os alunos imersos em atividades que destacam seus conhecimentos técnicos e habilidades de se trabalhar em equipe. Para que isso seja feito de forma efetiva em uma disciplina de ES, o professor precisará ter um planejamento bem definido para alcançar os objetivos e desenvolver habilidades específicas nos alunos.

4.1.5 Ameaças à Validade

A validade dos resultados do *survey* poderá ser comprometida se levada em consideração o nível significativo de não-respostas aos questionários. A fim de tratar este viés, e obter uma amostra representativa, buscou-se diversificar a quantidade de instituições participantes das diversas regiões do Brasil.

Mesmo levando em consideração que algumas pesquisas sugerem que alcançar maiores taxas de resposta não significa necessariamente resultados mais precisos (KROSNICK, 1999) e com o uso de amostragem probabilística, baixas taxas de resposta podem não implicar menor representatividade, este viés de amostragem poderá causar problemas em generalizar os resultados da pesquisa em âmbito nacional, como um todo, ou ainda em escala global. Porém, por conta do caráter heterogêneo e escala que o mercado de TI atende, os indícios e resultados encontrados aqui podem servir de parâmetro para maiores conjecturas.

Existe, ainda, um viés em perguntar ao estudante qual a sua aprendizagem, pois há muitos fatores que influenciam nessa sua percepção, como a afinidade com área de ES e o seu relacionamento com o professor; este último pode ser influenciado pelo posicionamento do professor diante dos alunos, pela sua organização na prática do ensino, utilização de recursos e relacionamento de forma geral. Outra variável discutida neste quesito refere-se ao posicionamento perante a própria matéria ensinada, voltada à dedicação, conhecimentos e planejamento dos assuntos e forma com que os assuntos serão tratados com os alunos (BORDENAVE; PEREIRA, 1985) (SANTOS, 2010).

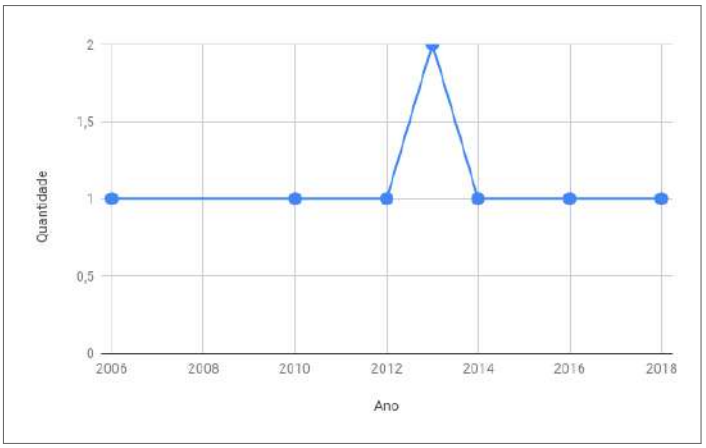
A fim de tratar esse viés, realizou-se a análise de validade de construção, que pode ser obtida por meio da correlação de um novo instrumento com um instrumento já validado. Nesse caso, analisou-se a correlação do questionário usado na coleta de dados sobre a importância dos tópicos com os questionários dos *surveys* conduzidos por (LETHBRIDGE, 1998) e (KITCHENHAM et al., 2005) que pode ser encontrado no Apêndice D.

Por fim, buscando garantir a validade interna, no que tange ao desenho de pesquisa para planejar e executar o *survey*, seguiram-se os *guidelines* definidos por (KITCHENHAM; PFLEEGER, 2008). Assim, pode-se garantir que as etapas necessárias foram seguidas na definição do design e na condução da pesquisa. Para embasar os resultados encontrados no *survey*, também foi feita uma revisão narrativa da literatura que será apresentada a seguir.

4.2 RESULTADOS DA REVISÃO NARRATIVA DA LITERATURA

A busca dos artigos que compuseram este estudo identificou 637 referências sobre uso de *frameworks* na educação de ES, das quais 8 publicações foram incluídas na revisão. Conforme apresenta na Figura 21, especificamente no que diz respeito aos 8 artigos selecionados, houve uma oscilação do número de estudos publicados por ano, variando entre o período de 2006 a 2018.

Figura 21 – Artigos por período



Fonte: Elaborado pela autora (2019).

As soluções destacadas nesta pesquisa indicam abordagens com iniciativas, práticas ou experiências voltadas para contribuir com a melhoria do ensino de ES a partir do uso de *framework*s.

A primeira solução é o *framework* HOT – humano, organizacional e tecnológico (HAZZAN; DUBINSKY, 2010) que atenta para a necessidade do ensino não abordar apenas os aspectos tecnológicos da engenharia de software, mas em vez disso, ele deve referir também ao ambiente de trabalho e ao quadro profissional. Neste sentido, a perspectiva do humano inclui aspectos cognitivos e sociais e refere-se à aprendizagem e interpessoais (colegas, clientes, gestão) processos. A perspectiva organizacional inclui aspectos gerenciais e culturais e refere-se à área de trabalho e questões que ultrapassam a equipe. A perspectiva tecnológica inclui aspectos práticos e técnicos e se referem a questões de how-to e relacionadas ao código.

O *framework* é aplicado em uma disciplina de ES que possui quatorze lições, organizadas em três iterações, permitindo assim rever os diferentes assuntos abordados várias vezes durante a disciplina bem como orientando o desenvolvimento de um produto de software. Os autores recomendam que a disciplina tome como base dois componentes principais, sendo eles: teórico e desenvolvimento de projeto. Esta estrutura permite que os alunos se familiarizem com conceitos de desenvolvimento de software, tanto a partir de uma perspectiva teórica e da perspectiva prática.

Para exemplificar o uso do framewok, os autores demonstram como as tarefas do HOT poderiam ser aplicadas na disciplina quando a lição fosse sobre Introdução ao Desenvolvimento Ágil de Software. A tarefa HOT seria: descreva o processo de desenvolvimento que você usou para o desenvolvimento do último projeto de software que você desenvolveu.

Os alunos são solicitados primeiro a descrever o processo de desenvolvimento que empregou em seu último projeto de software. Na segunda etapa, eles são solicitados a analisá-lo dentro do *framework* HOT. A motivação desta tarefa é permitir que os alunos sintam a relevância e aplicabilidade da estrutura HOT desde as fases iniciais da disciplina, e com relação à sua própria experiência em desenvolvimento de projetos.

A segunda solução a ser destacada é apresentada em (IBRAHIM; HALIM, 2014), onde o autor propõe um *framework* baseado em Project-Oriented Problem-Based Learning (POPBL). Este *framework* é projetado para os estágios básicos do ciclos de vida de desenvolvimento de software: análise, projeto, desenvolvimento e fases de teste. O autor afirma que a maioria das implementações de POPBL existentes destinam-se basicamente a um curso único e não são adaptados genericamente aos cursos de ES. Assim, a importância desse *framework* é aumentar potencial de reutilização da abordagem POPBL existente para uma implementação mais ampla e universal em diversos cursos de ES.

Neste estudo, o *framework* POPBL projetado possui três fases: início, execução e encerramento. Durante a fase de início, o contexto do projeto é definido em termos de construção de equipe e definindo problemas de estudo de caso. Primeiramente, uma introdução inicial da aproximação e dos conceitos de POPBL é explicada aos estudantes. Em segundo lugar, os alunos recebem um conjunto de perguntas de pesquisa.

O objetivo desta pesquisa é identificar os estilos e as características de aprendizagem dos alunos, o que é importante reconhecer na determinação de estratégias apropriadas que são necessárias para melhores melhorias. Em terceiro lugar, a formação da equipe é realizada em que cada equipe tem membros com critérios de gênero, raça e realização acadêmica (CGPA) escolhidos aleatoriamente. Os alunos de cada equipe têm que fazer *brainstorm* para o nome da equipe, o lema da equipe e também as regras do terreno da equipe. Isso é para induzir os espíritos de trabalho em equipe e para motivar outras habilidades suaves, como a comunicação e a cooperação entre os companheiros.

Finalmente, os instrutores divulgarão os detalhes dos problemas do estudo de caso do projeto. Os problemas do estudo de caso devem ser divulgados com base no ano atual de

estudos para os alunos de acordo com grupos. Por exemplo, uma vez que os estudantes PT1 são calouros que se inscreveram no primeiro semestre, assim, os problemas adequados e inadequados devem ser determinados pelo instrutor com base no *background* dos alunos e anos de experiências em ES.

A fase de execução é a fase mais importante do *framework* POPBL proposto. Durante esta etapa, os alunos se concentrariam na conclusão de seu projeto. Os alunos de cada equipe são obrigados a resolver problemas não estruturados e a decompor em partes gerenciáveis dentro dos estágios de ciclo de vida especificados. Vários problemas de estudo de caso diferentes devem ser analisados por cada equipe durante todo o semestre, juntamente com uma série de Marcos que precisam ser entregues para o projeto de estudo de caso.

Dentro deste período de tempo, a complexidade dos problemas no estudo de caso foi aumentada gradualmente para cada entregável. Isto é para garantir que a complexidade é mapeada para o conhecimento prévio dos alunos é sincronizado com o currículo curricular planejado e tópicos do curso. Durante a fase de encerramento, as tarefas envolvidas são necessárias na realização de *post-mortem*, revisão, e o curso de pesquisa de saída. Todas as pesquisas conduzidas são essenciais para analisar os resultados finais da adoção do POPBL nos cursos. O questionário pós-morte e de revisão destina-se a medir os três principais objetivos dos componentes do quadro POPBL, nomeadamente; i) aprendizagem cognitiva, II) aprendizagem colaborativa, e III) conteúdos; com um total de vinte e sete perguntas a serem feitas para avaliar o *feedback* dos alunos sobre esses três componentes.

A terceira solução encontra-se em (PEIXOTO et al., 2012), que usa a abordagem de jogos de simulação como uma forma de complementar o ensino de ES. Os autores criaram o FASENG (*framework* para jogos de simulação de Engenharia de Software) levando em consideração o conhecimento de engenharia de Software de jogos de simulação e seus requisitos comuns e a falta de apoio ao desenvolvimento de jogos de simulação.

FASENG é composto de três componentes (modelo de simulação, simulador e motor de simulação) que podem ser reutilizados na criação de novos jogos. Essa abordagem também traz resultados positivos, quando se pensa nos alunos como “nativos digitais”, ou seja, já cresceram usando computadores, telefones celulares e outros *gadgets*, tornando assim o uso do jogo uma forma natural de aprendizado.

Como quarta solução, (DING, 2013), propõe um *framework* para colaboração global no ensino de cursos de engenharia de software. O *framework* é desenvolvido com base em experiências e lições que o autor aprendeu em sua colaboração global no ensino de engenharia de software durante os últimos anos. Os alunos participantes do projetos-piloto foram os estudantes de ciência ou engenharia de software de computador graduação de Estados Unidos e outros países. Os alunos estudaram os princípios fundamentais de engenharia de software e práticas e colaboraram remotamente para produzir sistemas de software seguindo práticas de boa engenharia de software.

O *framework* proposto inclui diretrizes para a construção de uma infra-estrutura de colaboração eficaz, ideias para a concepção de um curso com alta colaboração e avaliação de procedimentos de qualidade para melhorar continuamente a qualidade de ensino. Desta forma, a importância deste *framework* está na oportunidade dos alunos poderem desenvolver colaborativamente projetos de software do mundo real em um ambiente que foi criado para esta finalidade.

Em quinto lugar, (OCHODEK, 2018), afirma que ensinar Scrum e outras práticas complexas de Engenharia de Software, métodos e ferramentas dentro de um curso acadêmico regular é muitas vezes uma tarefa desafiadora porque os exemplos mostrados aos alunos e seu ambiente de trabalho não são suficientemente realista. Como resposta a este problema, o autor propõe um *framework* centrado no Scrum que permite combinar palestras e aulas de laboratório com um projeto de conclusão minimalista, tudo dentro de um curso de ES regular.

O ponto focal do *framework* é a sincronização entre o conteúdo apresentado durante as palestras e aulas de laboratório com as iterações do *capstone projects* (sprints). O autor também compartilha sua experiência de 2 anos de condução de curso de Engenharia de Software organizado de acordo com a estrutura, juntamente com os resultados de uma pesquisa transversal avaliando as percepções dos alunos sobre a eficácia da abordagem. Para os alunos avaliados, 68% concordaram que o curso organizado usando o *framework* lhes permitiu aprender de forma mais eficaz os assuntos relacionados com Scrum ou outro conteúdo da ES. Também, em média, 67% por cento dos estudantes declarou que os mecanismos do *framework* tinham um impacto positivo na sua motivação para aprender.

A sexta solução de (HAZZAN; DUBINSKY, 2006), define um *framework* para o ensino de Métodos de Desenvolvimento de Software(MDS). Este *framework* define dez princípios conceituais e especifica práticas referentes as atividades de ensino que orientam os professores no processo de desenvolvimento da disciplina de ES e na avaliação dos estudantes. Os princípios são relacionados na Tabela 4.

Especificamente, o princípio I aborda a estrutura do curso; Princípios II – V abordam o ensino real do MDS; Princípios VI – IX abordam as pessoas envolvidas no ambiente educacional – os alunos e a equipe técnica; e o princípio X é um meta-princípio que sugere ação sobre pregação.

Há também o FRAMES (PORTELA; VASCONCELOS; OLIVEIRA, 2016), a sétima solução, um *framework* que foi estruturado de forma a ser aplicado a qualquer unidade de conhecimento da Engenharia de Software. A proposta do FRAMES se baseia em 4 (quatro) princípios derivados de trabalhos relacionados que abordam o uso de métodos focados no aluno no ensino de ES:

1. O estudante deve ser o foco do processo de aprendizagem;

2. A aprendizagem deve ser baseada na resolução de problemas;

Tabela 4 – Princípios do Framework para o Ensino de MDS

Num	Princípio	Breve descrição
I	Baseado em Projeto	A estrutura do curso é baseada em projeto e baseada em equipes.
II	Cognição	Quais aspectos devem ser enfatizados no ensino de MDS?
III	Ajuste	Ajuste dos MDSs para o framework no contexto de uma disciplina.
IV	Projeção	Projeção das noções de MDSs dentro de projetos de software.
V	Conectividade	O MDS no contexto do "mundo real" de desenvolvimento de software.
VI	Avaliação	Uma expressão de MDS na política de avaliação de estudantes.
VII	Ouvir	Ouvir as concepções, objetivos e expressões dos estudantes
VIII	Reflexão	Reflexão dos estudantes e progressos diários
IX	Coaching	Concepções dos professores, hesitações, objeções e expressões.
X	Inspiração	Inspirar o atual uso de MDS ao invés de impor.

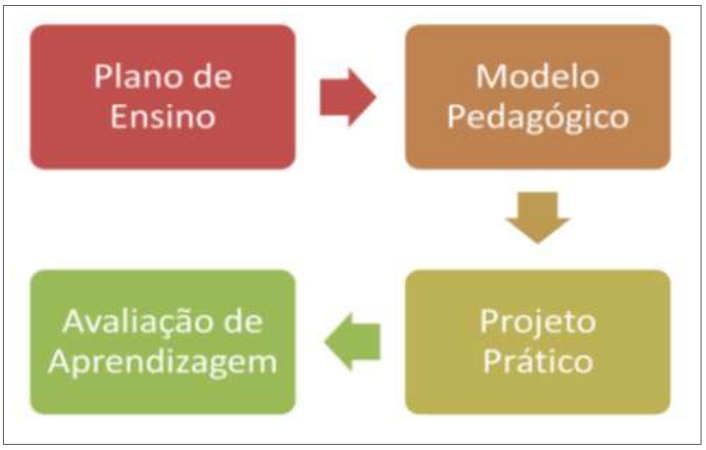
Fonte: Adaptado de (HAZZAN; DUBINSKY, 2006)

3. A realização de projetos práticos desenvolve competências e habilidades;

4. O modelo pedagógico deve seguir uma abordagem iterativa.

A partir destes princípios, definiram-se os módulos do FRAMES, conforme apresentado na Figura 22:

Figura 22 – FRAMES



Fonte: (PORTELA; VASCONCELOS; OLIVEIRA, 2016).

O módulo Plano de Ensino incorpora os tópicos das seguintes unidades de conhecimento: I) Engenharia de Requisitos; II) Processos de Software; III) Gerenciamento de Projetos de Software; IV) Projetos de Software; V) Verificação e Validação de Software; VI) Ferramentas e Ambientes. O módulo Modelo Pedagógico foi definido a partir de métodos de ensino focados nos alunos, abordagens iterativas de ensino, práticas de capacitação da indústria e estratégias de avaliação.

Já o módulo Projeto Prático sugere o uso de clientes/problemas reais, definição de prazos (cronograma), definição de papéis e responsabilidades pelos alunos. Para tal, disponibiliza um modelo de processo que define atividades,papéis e artefatos de acordo com as 6 unidades de conhecimento do Plano de Ensino.

Por fim, no módulo Avaliação de Aprendizagem, sugerem-se estratégias de avaliação que consistem na entrega de produtos de trabalho, participação individual dos alunos, questionários, dicionários de dados e mapas conceituais. Em (MERGEN et al., 2013), oitava solução, é apresentado um *framework* baseado em ciclos PDCA, onde os quatro estágios (Plan, Do, Check, Act) são adaptados ao contexto de ensino de disciplinas de engenharia de software. Os autores chamam essa adaptação de l-PDCA, onde l significa aprendizado.

Uma das principais premissas desta proposta é o uso de listas de verificação como instrumentos objetivos para avaliar os alunos. Com listas de verificação adequadas definidas, a avaliação pode ser feita de forma eficiente, alcançando resultados que sejam bem compreendidos por alunos e instrutores. No final de cada ciclo do PDCA, as listas de verificação também podem fornecer informações úteis sobre perfis existentes, como comportamento humano e fatores de desempenho. Além disso, os quatro estágios que envolvem a aplicação de listas de verificação possibilitam uma experiência de aprendizado uniforme, na qual o conhecimento é distribuído igualmente entre os alunos. A próxima seção destaca as percepções da autora desta pesquisa diante as soluções apresentadas e suas limitações.

4.3 DISCUSSÕES DA AUTORA

Durante a busca por modelos de *frameworks* para o ensino de ES, observou-se que a maioria das pesquisas apenas demonstra o que pode ser feito para melhorar seu ensino. Neste caso, há uma carência de soluções que possam indicar exatamente como o professor pode planejar uma disciplina de ES do início ao fim para trabalhar competências específicas no âmbito acadêmico com todas as suas dores, limitações e obstáculos. Para resumir as características gerais das pesquisa e justificar o uso do SELF, além da discussão apresentada nesta Seção, foi elaborada a Tabela 5.

A solução apresentada por (HAZZAN; DUBINSKY, 2010), visa capacitar os alunos para além de resolver problemas técnicos do mundo real. Busca trabalhar habilidades não-técnicas para cooperar e se comunicar com seus companheiros de equipe e pode ser aplicado em diferentes níveis de abstração de ES, já que a abstração é considerada como

tema principal em ambientes e processos de ES. A preocupação com a capacidade de um engenheiro de software lidar com problemas técnicos e não-técnicos pode ser vista também em mais duas soluções: (IBRAHIM; HALIM, 2014), onde os alunos são expostos a conhecimentos técnicos relevantes e habilidades não-técnicas tais como cooperação e comunicação eficaz e pensamento crítico e criativo; e em (DING, 2013), que ressalta a importância de um engenheiro saber como trabalhar com uma equipe diversificada e propõe um *framework* para colaboração com equipes que podem conter membros que estão localizados no mundo inteiro com diferentes origens culturais.

No entanto, nada adianta querer trabalhar habilidades nos alunos se o professor não possui uma forma clara e objetiva de montar sua disciplina para alcançar esse objetivo. O professor precisa ter um planejamento bem estruturado, pois sem isso as aulas irão se tornar insípidas e causará um aprendizado ineficaz nos alunos. Por isso o plano de ensino deve ser visto de forma estratégica, reflexiva, crítica e dinâmica, tendo que, no decorrer de sua execução, ser revisto, questionado e aperfeiçoado.

Em (PEIXOTO et al., 2012), usa-se a abordagem de jogos de simulação como uma forma de complementar o ensino de ES; em (OCHODEK, 2018) o autor incentiva os alunos a usarem a sprint atual do projeto para praticar o conteúdo aprendido durante as palestras e aulas de laboratório que foram vistos nas sprints anteriores; em (HAZZAN; DUBINSKY, 2006) os autores apresentam práticas que fornecem orientações práticas de como ensinar os métodos de desenvolvimento de software em capstone projetct; e em (MERGEN et al., 2013) o autor propõe um *framework* utilizando o ciclo PDCA para disseminar o conhecimento e avaliar a aprendizagem uniformemente através do desenvolvimento de um software.O uso dessas abordagem é interessante quando se leva em consideração que um dos maiores desafios do ensino da engenharia de software é fornecer experiência prática aos estudantes.

É importante trazer soluções práticas para os alunos, porém é mais importante saber trabalhar essas soluções de forma inteligente dentro da carga horária oferecida pela instituição de ensino. É preciso analisar de forma qualitativa como serão inseridos os conteúdos e as formas que eles serão trabalhados dentro da disciplina, para assim possibilitar que o processo de aprendizagem seja visto de forma mais direta.

Na solução apresentada por (PORTELA; VASCONCELOS; OLIVEIRA, 2016) os autores se preocupam apenas em mostrar o que pode ser melhorado no ensino de ES quando se utiliza o *framework* proposto. Além disso, no módulo de plano de ensino, os autores não cobrem todas as áreas de conhecimento de ES a fim de adequar a utilização do *framework* à limitada carga horária das instituições de ensino. Esta solução não guia o professor a fazer as melhores escolhas de conteúdo levando em consideração as mudanças frequentes no mundo da tecnologia.

Tabela 5 – Comparação entre as pesquisas

Característica/Pesquisa	**Melhoria no ensino de ES**	**Auxílio em como planejar a disciplina de ES**	**Habilidades técnicas**	**Habilidades não-técnicas**	**Atividades práticas**
(HAZZAN; DUBINSKY, 2010)	X		X	X	
(IBRAHIM; HALIM, 2014)	X		X	X	X
(PEIXOTO et al., 2012)	X		X		X
(DING, 2013)	X		X	X	X
(OCHODEK, 2018)	X		X	X	X
(HAZZAN; DUBINSKY, 2006)	X		X	X	X
(PORTELA; VASCONCELOS; OLIVEIRA, 2016)	X		X	X	X
(MERGEN et al., 2013)	X		X	X	X
SELF	X	X	X	X	X

Fonte: Elaborado pela autora (2019)

4.4 AMEAÇAS À VALIDADE

A revisão narrativa não exige um protocolo rígido para sua confecção; a busca das fontes não é pré -determinada, sendo frequentemente menos abrangente. A seleção dos artigos é arbitrária, provendo o autor de informações sujeitas a viés de seleção, com grande interferência da percepção subjetiva (CORDEIRO[1] et al., 2007).

Desta forma, a fim de reduzir este viés, buscou-se apresentar um objetivo claro da pesquisa que refletisse nos critérios de inclusão da revisão. Assim, houve uma maior probabilidade de as decisões sobre a inclusão de estudos serem realizadas de maneira mais rígida, evitando-se que fossem feitas com base em conhecimentos prévios sobre os resultados ou em características dos estudos a serem incluídos.

4.5 CONCLUSÕES DO CAPÍTULO

Este capítulo apresentou o método de pesquisa composto por um *survey* e uma revisão narrativa da literatura. A partir da realização do *survey*, identificaram-se as áreas de conhecimento mais adotadas em disciplinas de ES. Além disso, identificaram-se as estratégias de ensino e avaliação mais adotadas pelos professores. Observou-se também que embora a maior parte dos professores acreditem que a carga horária da sua disciplina seja suficiente, pela extensão dos assuntos abordados muitos não recebem a atenção necessária. Isso pode dever-se ao fato de que os contextos em que a engenharia de software se insere, seja difícil de trabalhar em sala de aula, o que torna as aulas teóricas de difícil entendimento e não desperta interesse nos alunos.

Apesar da dificuldade em trabalhar os assuntos de forma prática, em geral pelo menos quatro das áreas mais relevantes vistas em sala de aula, de maior aprendizado dos alunos e com necessidade no mercado estão alinhadas quanto se trata de conhecimento teórico e suas competências.

A revisão narrativa da literatura aclarou a premissa de que o uso de exemplos práticos para explicar conceitos e motivar a aprendizagem dos alunos parece ser o método mais eficiente de aumentar o aproveitamento da disciplina, mas há diversas formas de abordá-la, sendo a utilização de seminários e projetos as mais usadas. Esta observação embasa a escolha de usar as características do *Problem Based Learning* (PBL) como forma de aprendizado que estimula a pró-atividade, a definição de cronogramas e prazos e a atribuição de papéis e responsabilidades aos alunos.

Acredita-se que a análise dos resultados deste *survey* e da revisão é um passo para obter informações oportunas e valiosas em torno de um tema de interesse para a área de ES: como criar, para trabalhar no ambiente acadêmico, uma disciplina que foque nos tópicos considerados relevantes para a formação profissional dos alunos de ES. A partir dos resultados aqui mostrados, ao próximo capítulo apresenta a estrutura do *framework* proposto.

5 SELF: SOFTWARE ENGINEERING LEARNING FRAMEWORK

Este capítulo tem como objetivo apresentar a estrutura do SELF considerando seus elementos conceituais e componentes. A Seção 5.1 apresenta uma visão geral do framework. A Seção 5.2 mostra quais pré-requisitos necessários para se utilizar o *framework*. A Seção 5.3 destaca a relação de suas quatro etapas (PLAN, DO, CHECK e ACT) aos componentes metodológicos. Por fim, a seção 5.4 conclui o capítulo.

5.1 VISÃO GERAL

De acordo com o dicionário de engenharia de software (SOCIETY, 2010), um *framework* pode ser definido como um modelo que se refina ou especializa para fornecer partes de suas funcionalidades em determinado contexto.

Apesar de o conceito de *framework* estar mais ligado ao componente técnico para a construção de sistemas, o mesmo foi adaptado para adotar a perspectiva de um *framework* teórico que contempla uma estrutura hierárquica para a construção de uma disciplina de ES a partir de práticas que podem ser instanciadas de acordo com as necessidades específicas e distintas de cada contexto. A Figura 23 apresenta a visão geral do SELF com seus princípios norteadores.

Figura 23 – Visão Geral do SELF

Fonte: Adaptado de (IMAI, 1994).

1. **Conheça seu aluno**: É necessário levar em consideração o conhecimento anterior do aluno, ou seja, em quais disciplinas ele precisou passar para estar cursando a disciplina de ES e também quais disciplinas ele irá cursar ao longo do curso que poderão complementar os conteúdos ministrados na disciplina a ser lecionada. O SELF propõe que o aluno seja o foco do processo de ensino-aprendizagem e vê o

professor como um facilitador da aprendizagem criando condições para que os alunos aprendam.

2. **Planeje sua disciplina**: É necessário possuir uma forma clara e objetiva de montar a disciplina para alcançar os objetivo propostos para a mesma. O professor precisa ter um planejamento bem estruturado, pois sem isso as aulas irão se tornar insípidas e causará um aprendizado ineficaz nos alunos. Por isso o plano de ensino deve ser visto de forma estratégica, reflexiva, crítica e dinâmica, tendo que, no decorrer de sua execução, ser revisto, questionado e aperfeiçoado.

3. **Execute o processo de ensino**: O professor é um dos principais responsáveis pelo ensino, pesquisa e gerenciamento do processo educacional, é ele quem escolhe os métodos utilizados para captar a atenção do seu público em um determinado período. O professor deve utilizar instrumentos didáticos que priorizem a participação do aluno, gerenciando suas expectativas e habilidades (PRIKLADNICKI et al., 2009).

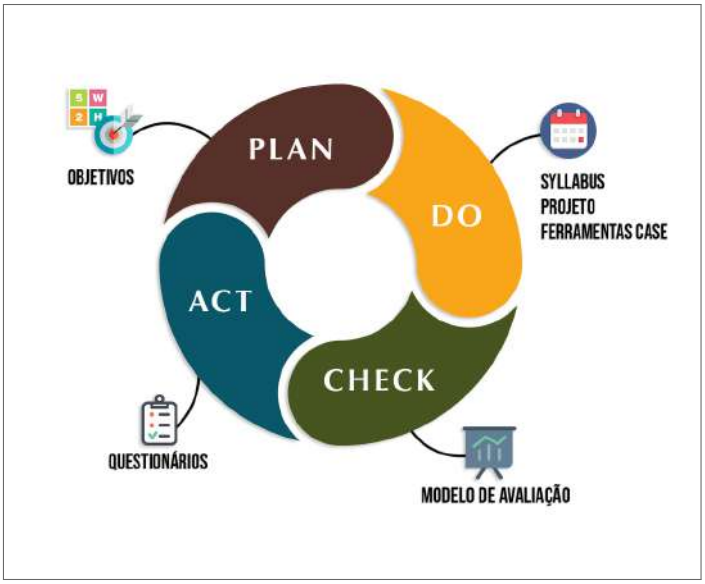
4. **Meça as competências trabalhadas nos alunos**: A avaliação da aprendizagem não deve acontecer apenas em um momento específico, e sim estar presente em todo o processo de ensino, integrando-se a todo contexto de aprendizagem, tomando formas distintas e avaliando o aluno como um contexto de competências diferenciadas.

5. **Não pare de melhorar**: A busca por melhores níveis de qualidade não é um desejo apenas da indústria de software, é também o objetivo dos profissionais da educação. Por isso, é importante que o professor busque melhorar continuamente seu processo de ensino-aprendizagem, levando em consideração a percepção dos seus alunos sobre a disciplina ministrada.

Já a estrutura conceitual do SELF é composta por técnicas e modelos, distribuídos por cada etapa do ciclo PDCA, que podem orientar como o processo de ensino-aprendizagem em ES é realizado, conforme ilustrado na Figura 24.

Para ES, um *framework* que se baseia nas etapas deste ciclo, faz menção ao gerenciamento de qualidade dos processos de ensino (Rodrigues; dos Santos, 2016). A Tabela 6 apresenta uma descrição do que será feito pelo professor em cada etapa do ciclo PDCA. Para o contexto educacional, como o processo de melhoria contínua é iterativo e cíclico, faz com que o professor se concentre em menos metas de cada vez e, posteriormente, se concentrar em outras, pode ajudá-lo a alcançar melhores resultados.

Figura 24 – Estrutura conceitual do SELF



Fonte: Elaborado pela autora (2019).

Tabela 6 – Atividades nos ciclos PDCA

Fase	**Descrição**
Plan	O professor realiza um planejamento que se refere a todas as metas que deseja atingir com a disciplina e essas metas são disseminadas entre todos os envolvidos através da divisão de tarefas
Do	O professor irá executar as tarefas estabelecidas no planejamento usando ferramentas de apoio ao ensino necessárias à realização dos objetivos
Check	Os alunos são avaliados conforme as estratégias de avaliação para medir seu aprendizado
Act	A disciplina é avaliada pelos alunos, e se algo não saiu conforme as expectativas originalmente formuladas, o professor busca corrigir o que não deu certo para que o problema não se repita

Fonte: Elaborado pela autora (2019)

5.2 PRÉ-REQUISITOS PARA USO DO SELF

Para usar este framework, espera-se que os professores tenham:

1. Formação em nível de graduação ou pós-graduação na área de Computação;
2. Interesse no uso de metodologias de ensino focadas no aluno;
3. Perfil de facilitador do processo de ensino-aprendizagem;

5.3 INSTANCIAÇÃO DO SELF

O SELF busca desenvolver competências e conhecimento nos alunos através de atividades de ensino. Estas atividades levam o aluno ao nível de conhecimento do conteúdo,

onde o mesmo deve lembrar dos conteúdos discutidos em sala de aula. Esse conhecimento pode ser utilizado nas atividades que possuem uma breve experiência prática, através do uso de ferramentas de apoio. Tais experiências permitem ao aluno ter a capacidade de entender o significado do conteúdo estudado. Outras atividades práticas permitem ao aluno aplicar o conhecimento adquirido. Dessa forma, o aluno terá a capacidade de aplicar o conhecimento adquirido em situações práticas. Nas subseções a seguir são descritas cada uma das etapas do framework.

5.3.1 Planejar

Tabela 7 – Ciclo de planejamento

Processo	Entradas	Saídas	Como	Quem
Planejamento da disciplina	5W2H Objetivo da disciplina	Plano de ação; Conteúdos que serão ministrados	Responder às perguntas da ferramenta 5W2H Definir as competências que serão trabalhadas nos alunos a partir do objetivo da disciplina	Professor

Fonte: Elaborado pela autora (2019)

O objetivo do planejamento é motivar o professor na construção de um efetivo planejamento para trabalhar a disciplina de forma eficaz dentro da carga horária estabelecida, atendendo às restrições e obrigações referentes a aulas práticas, teóricas e atividades complementares, por exemplo. A preparação introduz o professor para o problema em questão, descreve quais objetivos devem ser alcançados na disciplina e quais conteúdos vão resultar no alcance desses objetivos.

A Tabela 8 apresenta como a técnica 5W2H auxilia, ou guia, o professor nesta fase inicial de planejamento. A técnica 5W2H nada mais é do que um plano de ação especializado, estruturado e funcional, com etapas bem definidas. Em um ambiente dinâmico, como o da sala de aula, todas as atividades precisam ser bem delineadas para que não haja perda de tempo na ministração dos conteúdos (GOMES, 2014). Por essa razão a técnica 5W2H foi utilizada, para garantir que essa perda não ocorra e esclarecer completamente todas as possíveis questões que possam surgir, trazendo objetividade no processo de planejamento da disciplina.

Tabela 8 – Técnica 5W2H

Questões propostas pela 5W2H	**Elemento do problema**
What	Identificar e descrever o problema de forma adequada.
Why	Determinar o que será realizado, ou seja, qual a meta a alcançar.
Who	Identificar quem são os indivíduos associados ao problema.
Where	Determinar em qual lugar o problema em questão será enfrentado.
When	Determinar quando as tarefas propostas serão realizada e também a sua duração.
How	Estabelecer um plano para cada ação necessária para que a meta determinada no "Why" seja alcançada.
How much	Ajustar o orçamento à realidade financeira.

Fonte: Elaborado pela autora (2019)

No contexto da técnica 5W2H, o What se refere ao que será feito no primeiro momento em que se pensa em lecionar uma disciplina de ES. O Why justifica a necessidade de alinhar os objetivos da disciplina com as competências e habilidades que serão trabalhadas nos alunos. O Who reforça a necessidade do professor em trazer para o âmbito acadêmico (Where) as vivências reais que que vão complementar os assuntos puramente teóricos. Além de desenvolver uma ementa (How) para ser trabalhada durante o período da disciplina (When) levando em consideração a possibilidade de utilizar ferramentas pagas ou não (How much) como apoio ao processo de ensino e que leve o aluno a ter uma visão da ES que não seja apenas de desenvolvimento de software.

Ou seja, os alunos irão entender que ES não é somente a atividade de programar e conhecer linguagens e ferramentas de apoio à programação. Existem uma série de processos envolvidos que colaboram na "construção" de um produto de software, desde a especificação do projeto, seu planejamento de execução, desenvolvimento, testes, manutenção e evolução e, adicionalmente, talvez onde residem os principais riscos de um projeto de construção de software, os aspectos humanos (gestão de pessoas, comunicação, conflitos, entre outros). Portanto, ES claramente não se trata apenas de programação, uma atividade que pode ser desenvolvida de forma independente de outras pessoas, mas sim de um conjunto de atividades, tarefas e mais ainda, papéis que requerem trabalho em equipe e capacidade de comunicação e resolução de conflitos e ideias.

Depois de respondidas as questões da 5W2H o professor terá um norte para definir seu plano de ação de forma clara e objetiva que vai orientar o percurso que deve ser tomado na disciplina. Assim poderá partir para a escolha dos conteúdos que serão ministrados e que levarão ao alcance dos objetivos que o mesmo propôs para a disciplina. Em linhas gerais,

o objetivo da disciplina se decompõe em competências genéricas que serão trabalhadas pelo professor para capacitar os egressos da disciplina e por suas vez as competências determinam a necessidade de serem desenvolvidas em áreas de conhecimento que possuem conteúdos específicos para cada uma. Essa abordagem busca mitigar o problema de ser ter uma disciplina com conteúdos muito extensos, o professor deve dar ênfase à parte do conteúdo que irá capacitar o aluno nas competências pretendidas.

Para exemplificar como seriam as escolhas desses conteúdos, toma-se como base uma disciplina que tenha como objetivo: estudar, analisar, discutir, e aplicar os fundamentos de ES. A partir deste objetivo, a finalidade é capacitar o egresso nas seguintes competências:

1. Sugerir soluções de software para sistemas de informação
2. Produzir software para o atendimento de necessidades sociais ou organizacionais
3. Aplicar processos, técnicas e ferramentas de desenvolvimento de software
4. Desenvolver o compartilhamento de conhecimento, bem como de comunicação, negociação, colaboração e liderança

Para que o egresso da disciplina tenha essas competências bem desenvolvidas, precisa-se de conteúdos nas seguintes áreas de conhecimento de ES: Engenharia de requisitos (competência 1) - obter conhecimento sobre a elicitação, análise, especificação, validação e gestão de requisitos de software; Construção de Software (competência 1, 2) - onde o aluno aprenderá sobre o processo de construção de software; Testes de Software (competência 2, 3) - depois de construir o software, o aluno precisará verificar se o mesmo possui o comportamento esperado sobre um conjunto finito de casos de testes; Processos de ES (competência 3, 4) - despertar no aluno o interesse nas atividades de trabalho realizadas por engenheiros de software para desenvolver, manter e operar o software; Qualidade de Software (competência 2) - priorizar as características desejáveis de produtos de software e aos processos, ferramentas e técnicas utilizadas para alcançar essas características; e Métodos e modelos de ES (competência 1, 2, 3 e 4) - fornecer uma abordagem para a especificação sistemática, concepção, construção, teste e verificação do software final.

Assim, o professor poderá montar seu plano de ação conforme apresentado na Tabela 9. Com seu plano de ação e os conteúdos que serão ministrados, o professor segue para a próxima etapa do ciclo.

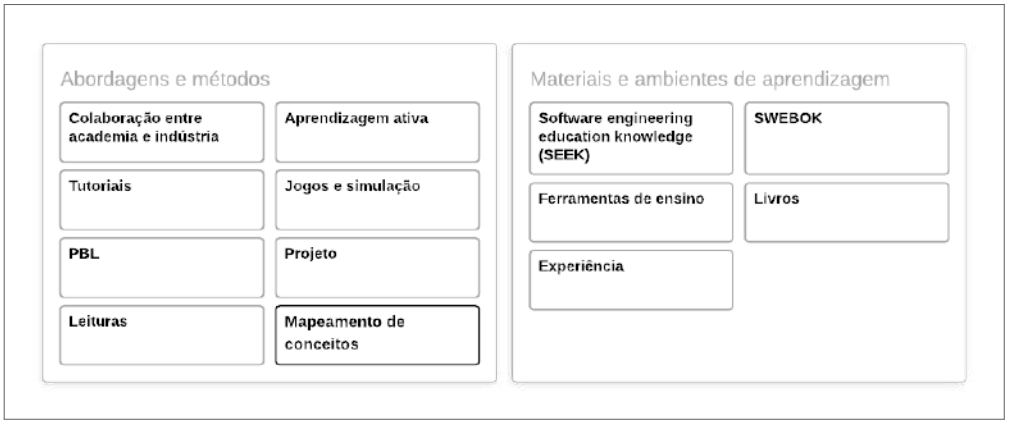
Nesta etapa, o professor também precisará levar em consideração como se dará o seu processo de ensino, e para isto, o SELF oferece uma representação de abordagens e métodos e materiais e ambientes de aprendizagem que podem ser utilizados, conforme pode ser visto na Figura 25.

Tabela 9 – Plano de ação

Meta	Porque trabalhar este assunto	Ações
Definições de requisitos de software.	Dizem respeito a materialização de uma necessidade ou solicitação realizada por um software. Para descobrir as finalidades desse software, precisam haver reuniões com as partes interessadas, porém há uma série de dificuldade que podem surgir destas reuniões. Os objetivos do cliente quanto ao software podem não ficar explícitas ou podem ser difíceis de desenvolver, o que pode impactar negativamente na satisfação das partes interessadas. Por isso é de grande importância que o aluno saiba como identificar as partes interessadas e suas necessidades, e documentando-as em uma forma que seja capaz de análise, comunicação e subsequente implementação (NUSEIBEH; EASTERBROOK, 2000)	1. Analisar problemas do mundo real e organizá-lo em requisitos e casos de uso 2. Apresentar diferentes técnicas e métodos de levantamento de requisitos e utilizar um método de levantamento de requisitos. 3. Orientar os alunos para a implementação de um software de acordo com o levantamento de requisitos

Fonte: Elaborado pela autora (2019)

Figura 25 – Representação de abordagens, métodos, materiais e ambientes de aprendizagem



Fonte: Elaborado pela autora (2019).

5.3.2 Executar

Tabela 10 – Ciclo de Execução

Processo	Entradas	Saídas	Como	Quem
Execução do processo de ensino	Conteúdos Carga horária	Plano de aula e atividades de ensino	Definir plano geral para a disciplina Definir ordem de ensino dos conteúdos Definir atividades de ensino	Professor e monitores

Fonte: Elaborado pela autora (2019)

Nesta etapa, o professor irá escolher a ordem dos conteúdos que serão ministrados levando em consideração a carga horária da disciplina e deverá montar seu plano de aulas. Levando isso em consideração, para as atividades de ensino, além das aulas expositivas, o SELF propõe atividades de monitoria, estudos independentes e realização de projeto como pode ser visto na Tabela 11.

Tabela 11 – Atividades gerais do plano de aula

Atividades	Responsável	Periodicidade
Aulas expositivas	Professor	Semanalmente
Monitorias	Monitores da disciplina	1 a 2 vezes por mês
Estudos independentes	Alunos	Sempre que necessário
Projeto	Alunos	Acompanha a quantidade de iterações

Fonte: Elaborado pela autora (2019)

As atividades devem ocorrer de forma combinada. Os estudos independentes e as atividades do projeto serão feitas de acordo com os conteúdos ministrados nas aulas expositivas e monitorias. Logo, se o professor ministra conteúdos relacionados à teste de software e na monitoria os alunos são apresentados a ferramentas de teste, estas devem ser usadas no desenvolvimento do projeto e estudos independentes devem ser feitos para sanar possíveis dúvidas.

Esse plano foi criado com o objetivo de combater alguns desafios do ensino de ES. A quantidade de aulas expositivas, que se referem ao formato de aula formal onde o professor apresenta os conteúdos de ES aos alunos em sala de aula, deve ser pensada para não deixar a disciplina puramente teórica e assim causando o desinteresse dos alunos por determinados assuntos. Como o professor já fez as escolhas dos conteúdos específicos na etapa anterior, ele pode alocar esses conteúdos para serem trabalhados tanto em sala como durante a realização do projeto e monitoria dentro da carga horária da disciplina.

Para mitigar o problema de dificuldades em conciliar teoria e prática ou desenvolver a parte prática da disciplina, foram incluídos no plano, aulas de monitoria e projeto que

buscam trazer para o ambiente acadêmico atividades onde os alunos poderão aplicar os conhecimentos teóricos. Monitorias são aulas práticas onde os alunos são apresentados às ferramentas que serão utilizadas durante a disciplina. Estudos independentes serão realizados pelos alunos durante o desenvolvimento do projeto para solucionar possíveis problemas, levando-os assim a estudar sobre possíveis assuntos emergentes em ES e levando esse conhecimento para os alunos em sala de aula.

O projeto será atividade prática onde os alunos poderão aplicar os conhecimentos teóricos vistos em sala de aula. Haverá a iteração inicial onde que começa com a definição dos times e a escolha de um cliente real. Se não houver um cliente real, o professor (ou algum outro membro responsável pela condução da disciplina como, por exemplo, estagiário docente ou professor auxiliar) assumirá esta posição. Depois da escolha de grupos e temas, haverá uma reunião com o cliente onde será decidido o escopo do projeto e a especificação dos requisitos. As próximas atividades serão de criação do planejamento do desenvolvimento de software.

Haverá também iterações intermediárias, onde o time irá iniciar o desenvolvimento do projeto que compreende a realização das atividades planejadas na iteração anterior. Cada iteração terá um histórico das atividades que deverão ser realizadas. Ao fim de cada iteração, haverá uma apresentação em sala do que foi realizado e o que poderia ser melhorado para a próxima iteração. Durante esta fase, os conhecimentos adquiridos a partir dos estudos independentes vão guiar o time na resolução de problemas encontrados. O cliente poderá pedir mudança de alguns requisitos ou excluir/adicionar funcionalidades. O time precisará trabalhar o gerenciamento do tempo para entregar as funcionalidades no prazo. Em seus estudos independentes o aluno poderá aprender sobre especificação, design, arquitetura e teste de software.

Por fim, a iteração final, onde o time deve finalizar o projeto e se preparar para a apresentação final, onde o produto irá ser exposto para os colegas e cliente. Além disso, propõe-se a entrega de um *postmortem* contendo todas as atividades realizadas desde o início do projeto até o seu encerramento, promovendo uma reflexão sobre o processo de criação realizado durante o semestre. Em seus estudos independentes o aluno poderá obter conhecimento sobre entrega de software.

Após a escolha das atividades de ensino, o professor irá montar seu *syllabus* e executar o processo de ensino. Para haver uma padronização, é importante que as tarefas que são comuns a todas as disciplinas devam ser as primeiras a serem inseridas no *syllabus* como: apresentação da disciplina, revisão do conteúdo e prova. O professor deve considerar que a ES é uma disciplina onde há muita teoria, e para que os alunos não sejam sobrecarregados com vários conteúdos em uma única prova, a divisão entre quantidade de conteúdo por prova deve ser levado em consideração, por isso, há a necessidade de escolher o dia adequado para a realização da prova escrita.

5.3.3 Checar

Tabela 12 – Ciclo de checagem

Processo	Entradas	Saídas	Como	Quem
Analisar o progresso e desempenho dos estudantes	Estratégias de avaliação	Instrumentos para avaliação de grupo/ individual, conteúdo e satisfação do cliente; Resultados da avaliação	Definir estratégias de avaliação Aplicar as estratégias de avaliação Resumir os resultados da avaliação Apresentar os resultados aos estudantes e orientá-los quanto às perspectivas coletiva e individual	Professor, monitores e clientes

Fonte: Elaborado pela autora (2019)

Como é importante não haver apenas uma forma de avaliação de conhecimento, o aspecto-chave deste *framework* é que o panorama de avaliação é composto por uma coleção de atividades de avaliação que cobrem três dimensões: (1) avaliação de grupo/individual, (2) conteúdo e (3) satisfação do cliente. Detalhes destas dimensões podem ser vistas abaixo:

- **Avaliação de grupo/individual**
  - Apresentação de Relatório: Nessas apresentações o professor avalia o progresso do grupo e as contribuições de cada indivíduo. Ao fim das apresentações o professor oferece um *feedback* oral sobre suas percepções do projeto e sugere melhorias.
  - Artefatos: Os artefatos do projeto são avaliados quanto a sua qualidade e seu tempo de entrega (artefatos entregues além do prazo são penalizados). Os alunos podem solicitar *feedback* de seus artefatos antes da entrega final do mesmo, para que suas versões intermediárias possam ser melhoradas.
  - Entrega Final: Verificar os resultados de cada equipe de projeto ao considerar (1) Apresentação de relatório; (2) Planejamento das Atividades; (3) Contribuição individual; e (4) Andamento do projeto na data final.
  - Avaliação 360: Ao final do projeto, os alunos devem avaliar seus colegas de equipe quanto à proatividade, eficiência, trabalho em equipe e comunicação. Nessa avaliação os alunos terão uma quantidade x de pontos que deverão ser

distribuídos entre si de acordo com a avaliação. Os alunos precisarão entrar em acordo quanto a pontuação final de cada um, e se isso não ocorrer, o professor mediará esse conflito até que se encontre um resultado comum. A quantidade total de pontos será a soma das notas dos artefatos das iterações do projeto. Exemplo: Um grupo de 4 alunos irá ter uma quantidade de pontos máxima de 40 se todos os artefatos forem entregues conforme o estabelecido.

- **Conteúdo**: Para esta dimensão, serão feitas provas objetivas durante a disciplina para avaliar se o aluno pode identificar princípios de ES e oferecer soluções de ES para pequenos estudos de caso.
- **Satisfação do cliente**: Por fim, o cliente avalia a qualidade da solução proposta, levando em consideração se as reuniões com o grupo foram úteis, se houve uma boa compreensão do problema, nível de planejamento das atividades e clareza nas apresentações. Estes critérios serão classificados de acordo com uma escala de 1 a 5, onde "1 - Insuficiente"; 2 - "Satisfatório"; "3 - Bom"; "4 - Muito Bom; "5 -Excelente".

5.3.4 Agir

Tabela 13 – Ciclo de ação

Processo	Entradas	Saídas	Como	Quem
Avaliar a execução do processo	Questionário	Resultados do questionário	Definir os marcos da avaliação do processo; Apresentar os objetivo da avaliação; Aplicar o questionário; Sintetizar e avaliar os resultados; Discutir sobre melhorias ao processo de ensino; e Aplicar estratégias para melhorias do processo de ensino	Estudantes

Fonte: Elaborado pela autora (2019)

Para a última etapa, o professor irá aplicar um questionário aos alunos da disciplina para que a mesma seja avaliada em diversos aspectos que dizem respeito a atividade do professor. Com base nas dimensões do ensino de (BENTON; CASHIN; KANSAS, 2012), foi

elaborado um formulário de avaliação do ensino para os alunos, que mede as seguintes dimensões:

- **Itens globais e design da disciplina**: As questões desta dimensão são projetadas para servir como indicadores de pontos fortes e fracos gerais e são úteis para o processo de melhoria. Utilizam escala de pontos que vai de 1 a 6, sendo 1- Discordo muito fortemente 2-Discordo fortemente 3-Discordo 4-Concordo 5-Concordo fortemente 6-Concordo muito fortemente.
- **Ambiente de ensino e disponibilidade do(a) professor(a)**: As questões desta dimensão são projetadas para captar a visão dos(as) alunos(as) sobre atividades e comportamentos em sala de aula mais específicos. Ajudam o(a) professor(a) a encontrar pontos que precisam de melhoria e só fazem sentido quando visto no contexto individual da sua disciplina. Utilizam escala de pontos que vai de 1 a 6, sendo 1-Nunca 2-Raramente 3-Ocasionalmente 4-Geralmente 5-Frequentemente 6-Sempre.
- **Avaliação do aprendizado**: Questões de conceito geral que visam avaliar o impacto das atividades instrucionais e do comportamento na aprendizagem dos(as) alunos(as). Utilizam escala de pontos que vai de 1 a 6, sendo 1-Nunca 2-Raramente 3-Ocasionalmente 4-Geralmente 5-Frequentemente 6-Sempre.
- **Itens descritivos**: Questões abertas que buscam captar a visão dos(as) alunos(as) sobre os pontos fortes, fracos e melhorias que poderiam ser feitas na disciplina.

Com as respostas dos questionários o professor terá a consciência dos problemas levantados e poderá entender sua natureza, seu contexto, as potencialidades e as limitações do mesmo. Em seguida poderá partir para a resolução dos problemas onde é importante o professor registrar um *postmortem* sobre o que deu certo na sua metodologia, o que não deu e porque não deu e quais mudanças serão aplicadas. Assim terá conhecimento se precisará passar por todo o ciclo PDCA novamente ou apenas reinventar sua metodologia voltando sua atenção à fase Do. É importante também que o professor use as boas práticas de ES na própria disciplina, considerando cada execução da mesma como um *release* e se preocupando com rastreabilidade e versionamento dos itens de configuração gerados.

5.4 CONCLUSÕES DO CAPÍTULO

Este capítulo apresentou um *framework* de apoio ao ensino de ES que possui como principal diferencial guiar o professor em como criar uma disciplina que integre teoria e prática, levando em consideração o conhecimento prévio dos alunos. O professor também usará práticas de ensino classificadas como humanistas, pois são focadas no aluno.

Inicialmente, definiram-se as etapas do *framework* apresentadas neste capítulo, derivadas dos resultados obtidos na execução dos métodos de pesquisa. A partir da revisão da literatura e do survey com professores, alunos e profissionais identificou-se a importância das diversas áreas da ES na visão de cada um dos públicos-alvos; qual a ênfase educacional dada para estas áreas nos cursos de graduação em informática e quais áreas e competências os profissionais de ES consideram mais importantes para a profissão, levando em consideração a demanda do mercado.

O Capítulo 6 apresenta a aplicação e avaliação do *framework*, que foi feita a partir de um painel de especialistas.

6 APLICAÇÃO E AVALIAÇÃO DO SELF

Este capítulo tem como objetivo apresentar a aplicação do SELF, na Seção 6.1 que foi realizada na disciplina de Engenharia de Software do Centro do Informática da Universidade Federal de Pernambuco. A seção 6.2 apresenta a avaliação do SELF através do método painel de especialistas. Por fim, a seção 6.3 apresenta as conclusões do capítulo.

6.1 APLICAÇÃO DO SELF

6.1.1 PLAN - Planejamento da disciplina

Como a primeira entrada deste ciclo é a técnica 5W2H, o professor iniciou seu planejamento, respondendo as questões abordadas pela técnica, levando em consideração o contexto da disciplina que é no entendimento do cenário da indústria de software. A Tabela 14 apresenta como a técnica foi aplicada.

Tabela 14 – 5W2H na disciplina IF977

Questões propostas pela 5W2H	**Elemento do problema**
What	Planejar uma disciplina onde os(as) alunos(as) deverão adquirir as principais competências que um engenheiro de software deve possuir e aplicar a teoria em um projeto com clientes e situações reais ou bem próximas disso
Why	Para garantir que os objetivos educacionais de estudar, analisar, discutir e aplicar os fundamentos de ES através de um projeto de sistema de informação SaaS simples, estejam alinhados com as competências e habilidades de um engenheiro de software
Who	Professor(a), professores(as) auxiliares, estagiários docentes e monitores
Where	Sala de aula, laboratórios de pesquisa, salas de reunião, visitas a centros de pesquisa, de treinamento, empresas e instituições governamentais
When	Durante todo semestre letivo
How	Criando uma ementa que associe meus objetivos pessoais para a disciplina com os tópicos fundamentais da ES e vinculados com as eventuais demandas do Arranjo Produtivo Local, Nacional ou Global
How much	Não se aplica

Fonte: Elaborado pela autora (2019)

As competências escolhidas para serem trabalhadas nos alunos tiveram sua classificação atribuída aos níveis da taxonomia de Bloom (BLOOM et al., 1956) e são apresentadas na Tabela 15.

Tabela 15 – Classificação das competências para a disciplina IF977

Competência	Classificação
Sugerir soluções de software para sistemas de informação	Conhecimento
Produzir software para o atendimento de necessidades sociais ou organizacionais	Compreensão
Aplicar processos, técnicas e ferramentas de desenvolvimento de software	Aplicação
Desenvolver o compartilhamento de conhecimento, bem como de comunicação, negociação, colaboração e liderança	Aplicação

Fonte: Elaborado pela autora (2019)

Com base nos conteúdos fornecidos pelo SWEBOK que trabalham as competências escolhidas, o professor escolheu os conteúdos que seriam ministrados:

- **Engenharia de requisitos**: Fundamentos da engenharia de requisitos; requisitos funcionais e não funcionais
- **Construção de Software**: Fundamentos de construção de software; reuso; programação em par; revisão de código
- **Testes de Software**: Fundamentos de testes; técnicas de teste; processos de teste
- **Processos de ES**: Ciclo de Vida
- **Qualidade de Software**: Validação e verificação de software
- **Métodos e modelos de ES**: Metodologias ágeis; TDD e BDD

Para montar seu plano de ação, o professor justificou suas escolhas levando em consideração a relevância de alguns conteúdos para a academia e indústria, conforme apresentado na Tabela 16.

Tabela 16 – Plano de ação IF977

Meta	Porque trabalhar esse assunto	Ações
Programação em par	Esta técnica de programação é uma ótima ferramenta para professores que desejam melhorar a aprendizagem dos alunos em relação ao desenvolvimento de software. Ela facilita a produção de código de alta qualidade fazendo com que dois indivíduos trabalhem na codificação do mesmo software no mesmo computador ao mesmo tempo, promovendo assim a comunicação, captura erros mais rapidamente, e tem como resultado final um código de maior qualidade no mesmo ou menos tempo do que as técnicas mais tradicionais (BECK; GAMMA, 2000)	1. Apresentar o conceito de programação em par e suas vantagens e desvantagens 2. Realizar exercícios em duplas, durante as monitorias para que os membros compartilhem o conhecimento agregado no código produzido

Continua na próxima página

Tabela 16 – Continuação

Meta	Porque trabalhar esse assunto	Ações
Técnicas de teste	Erros em softwares podem ser caros ou até mesmo perigosos. Por isso, é importante que os alunos saibam como identificar a exatidão, integridade e qualidade do software que será desenvolvido através de técnicas de testes para garantir que o software esteja livre de defeitos. Porém, os testes não podem ser usados apenas para corrigir os erros envolvidos no software, eles precisam se concentrar em maximizar a satisfação do cliente. O teste não cria qualidade em um produto, mas a qualidade não pode ser obtida sem um teste completo. As atividades de testes estão inseridas nas atividades de garantia de qualidade de software que procuram manter a qualidade do produto que está sendo desenvolvido, fazendo com que o produto final seja portador de vários parâmetros que caracterizam sua qualidade (FABBRI; FERRARI; CAMARGO, 2014). Testes verificam se as metas de qualidade predeterminadas pelos clientes foram atendidas. Assim, os testes são capazes de avaliar ou validar o software desde antes de existir até sua implantação para o cliente, assegurando assim a entrega de valor. Técnicas de teste como (*Behavior-Driven Development - BDD*) auxiliam na garantia da qualidade no atendimento às necessidades do cliente.	1. Apresentar os conceitos de técnicas de teste; técnicas funcionais de teste 2. Incentivar que os alunos utilizem alguma técnica apresentada na realização do projeto

Continua na próxima página

Tabela 16 – Continuação

Meta	Porque trabalhar esse assunto	Ações
Metodologias ágeis	Enfatiza o TDD (*Test-Driven Development*, que reduz erro e supre a necessidade da indústria de se escrever melhores testes; também foca em história de usuários para validar os requisitos com o cliente e velocidade para medir o progresso. Na metodologia ágil a equipe de desenvolvimento faz novas versões do software a cada duas semanas. As equipes trabalham basicamente na refatoração de código contínuo durante o tempo de vida do software. As equipes são pequenas e trabalham em iterações que entregam protótipos incompletos a cada iteração. Essa abordagem pode auxiliar na criação de uma programação produtiva para trabalhar os assuntos da disciplina dentro dos limite tempo da disciplina (FOX; PATTERSON, 2012).	1. Apresentar o conceito de metodologia ágil 2. Apresentar as principais metodologias ágeis 3. A parte prática deste conteúdo será trabalhada durante a execução do projeto, onde os alunos irão utilizar a metodologia SCRUM

Fonte: Elaborado pela autora (2019)

Para além dos conteúdos mostrados na Tabela 16 , e levando em consideração as mudanças frequentes no mundo da tecnologia (FOX; PATTERSON, 2012), o professor decidiu incluir os seguintes conteúdos por serem de grande relevância para a indústria. Estes conteúdos e em quais cenários eles são importantes são:

- **Arquitetura Orientada a Serviço (SOA)**: A arquitetura de software é um dos principais fatores que proporcionam ganhos efetivos em agilidade e eficiência na manutenção e evolução dos sistemas de informação corporativos, fator predominante em ambientes competitivos. Como resultado disto, muitas organizações passaram a reavaliar a função da tecnologia da informação na formação de suas estratégias de negócios. Atualmente, o projeto de arquitetura preferido para fornecer agilidade organizacional, promover a adaptabilidade e interoperabilidade do sistema é a SOA (NIKNEJAD; AMIRI et al., 2019) (ERL, 2016).

- **Computação em Nuvem e SaaS**: A computação em nuvem com sua capacidade de fornecer acesso sob demanda a plataformas de software atraiu o interesse da indústria. Esse modelo facilita com que as empresas expandam as operações em um ritmo mais rápido e distribuam produtos e serviços de maneira econômica, porque pagam pelo uso real, sem custos iniciais e escalonáveis. Com a facilidade proporcionada pela modalidade SaaS (*Software as a Service*) - que tem o potencial de fornecer oportunidades substanciais para as organizações melhorarem suas tecnologias de informação sem preocupações com custos e gerenciamento - as empresas estão mais propícias a mudar os tipos de plataformas utilizadas para alinhá-las às estratégias do negócio (SAFARI; SAFARI; HASANZADEH, 2015).

A partir desta especificação dos conteúdos a serem ministrados, o professor passou para a próxima etapa do ciclo, onde começou a haver, de fato, a estruturação da disciplina.

6.1.2 DO - Execução do processo de ensino

O futuro do software dito tradicional foi revolucionado com o surgimento da Computação em Nuvem, Software como Serviço (SaaS), Arquitetura Orientada a Serviço (SOA) e Metodologias Ágeis que trouxeram ferramentas e técnicas que podem ser melhores usadas em sala de aula e assim facilitar a forma de ensino (FOX; PATTERSON, 2012). Os alunos podem desenvolver e implantar suas aplicações SaaS na nuvem sem usar recursos do campus que muitas vezes estão sobrecarregados.

A metodologia ágil trouxe técnicas modernas, métodos e ferramentas (i.e. frameworks) de apoio programação ágil como o *Ruby on Rails*, que fornece produtividade por meio da reutilização extensiva de software (FOX; PATTERSON, 2012), metaprogramação, concisão, síntese e convenção sobre configuração.

Com o uso desses recursos o professor pode fazer com que os alunos vivenciem o ciclo de vida de todo o software dentro do ambiente acadêmico, fazendo-os ter a experiência de criar softwares mais rapidamente e alinhado ao que os clientes precisam e o mercado prega. Por isso, decidiu-se adotar o uso de um projeto, no formato de uma fábrica de software e ferramentas de desenvolvimento aderentes aos preceitos e manifesto ágil durante a disciplina como, por exemplo, uma abordagem baseada em *scrum* para a gestão dos projetos que foram executados pelos alunos.

Nestes projetos, algumas práticas do *scrum*, como o desenvolvimento iterativo e incremental baseado em iterações, foram integradas ao ambiente acadêmico, para que a aprendizagem pudesse acontecer de forma iterativa e incremental. O professor organizou os conteúdos ministrados para que as entregas das iterações fossem bem-sucedidas e de acordo com o que os alunos estavam praticando no projeto. Assim, os alunos puderam aprender e praticar os fundamentos de ES e usá-los quando forem para a indústria.

Como os conteúdos já foram definidos no ciclo anterior, o professor precisou separar a carga horária para cada atividade de ensino, com isso, o plano geral para IF977 ficou definido conforme mostra a Tabela 17.

Tabela 17 – Carga horária do plano geral

Atividades	**Carga Horária**
Aulas Expositivas	32h
Monitorias	12h
Estudos independentes	48h
Projeto	56h

Fonte: Elaborado pela autora (2019)

Levando em consideração a carga horária de 60h e os conteúdos que serão ministrados, foi definido o *syllabus* para IF977 que pode ser visto aqui.

O professor escolheu que a maioria das aulas expositivas fossem dadas do início para o meio do semestre, começando com 5 aulas expositivas e 1 monitoria. Já no primeiro dia de aula o professor passou as informações de como ocorreria a realização do projeto e das avaliações e encorajou os alunos a iniciarem a definição dos times. Após o meio do semestre, a quantidade de aulas expositivas diminuiu, passando a ser 2 aulas expositivas e 1 de monitoria. Como o conteúdo para as aulas expositivas já haviam sido lecionados, ao final do semestre o professor também deixou 4 aulas práticas para que os alunos pudessem trabalhar exclusivamente no projeto. Durante todo o semestre os alunos precisaram realizar estudos independentes para sanar possíveis dúvidas, e se as mesmas ainda persistissem, os alunos poderiam buscar ajuda do professor ou monitores A primeira prova foi realizada na 7° semana de aula com questões sobre 6 conteúdos e a segunda prova foi realizada na 15° semana com questões sobre 5 conteúdos.

Para a realização do projeto, os alunos foram divididos em equipes de no máximo seis componentes. A intenção do projeto foi que as equipes desenvolvessem, preferencialmente, projetos voltados para beneficiar a sociedade, de alguma forma, seja para organização vinculadas ao campus da UFPE, organizações sem fins lucrativos, para construir uma solução voltada para melhorar a qualidade de vida dos cidadãos.

Houve formação de 6 equipes para a realização dos projetos. Os alunos ficaram livres para escolher quem seriam os integrantes de cada projeto, sendo o número máximo de 6 integrantes. A seguir pode-se encontrar o nome de cada projeto com uma breve descrição do mesmo e o perfil dos integrantes:

1. Acessible Life: Objetivando melhorar a vida de pessoas portadoras de deficiência ou com mobilidade reduzida, o Accessible Life é um aplicativo para registro de denúncias referentes à acessibilidade. Também servindo para auxiliar seus usuários com um ranking de empresas mais denunciadas em relação a acessibilidade e um mapa interativo com todas as denúncias feitas pelos próprios usuários. Para as empresas, são designados perfis para comentar as denúncias e resolvê-las. 5 alunos, sendo todos do sexo masculino.

2. Propes: O Propes é uma aplicação mobile com o intuito de armazenar dados de testes realizados em atletas e os disponibilizar. O aplicativo tem como objetivo armazenar dados de atletas que sejam relevantes à uma análise física. Esses dados serão dispostos de uma maneira simples e que deixe o processo de analisar resultado de testes realizados nos atletas muito mais rápido e intuitivo. Com três níveis de usuário o aplicativo é capaz de mostrar apenas as informações relevantes para cada nível, os atletas poderão ver seus desempenhos e avanços, os técnicos poderão analisar seus times como um todo, além de cada atleta individualmente e os avaliadores físicos poderão fazer uma análise mais profunda tendo acesso aos dados de cada atleta avaliado por eles. 4 alunos, sendo todos do sexo masculino.

3. Iggle: Destinado às pessoas interessadas em um estilo de vida mais saudável, o Iggle é um aplicativo que permite aos seus usuários tanto prover serviços no ramo de saúde fitness como contratá-los, tudo em um ambiente funcional para cada tipo de interesse. Para o profissional, é disponibilizada a criação de um perfil com suas informações e serviços e um canal de comunicação com os clientes interessados. Aos clientes, o aplicativo permite uma análise desses profissionais e serviços e um processo de marcação de consultas mais enxuto, que atenda aos seus critérios. Além disso, é oferecido também um sistema de avaliação pós-consulta entre cliente-profissional, promovendo uma garantia de qualidade para todos os interessados. 5 alunos, sendo 3 do sexo feminino e 2 do sexo masculino.

4. MedMapper: A plataforma MedMapper tem como principal objetivo reunir informações em um só lugar sobre as unidades de saúde da região metropolitana do Recife e os tratamentos que elas oferecem relacionados ao Zika e a microcefalia, com o intuito de facilitar a busca das mães de crianças que tiveram seus filhos afetados por essas doenças. 4 alunos, sendo todos do sexo masculino.

5. X-System: O sistema PontoWeb da Universidade Federal de Pernambuco é responsável pelo controle da frequência dos Técnicos Administrativos. Com migração do Sig@ para SIGAA o PontoWeb irá passar apenas para a manutenção e parte técnica do sistema. Então será preciso fazer uma transposição e transferência contínua dos dados de frequência dos funcionários do sistema PontoWeb para o SIGAA, onde será desenvolvido e implantado um software responsável e com a capacidade de realizar essa tarefa. 4 alunos, sendo todos do sexo masculino.

6. Pet helper: O Pet helper é uma aplicação web que funciona como uma ferramenta para donos de abrigos de animais. Nessa ferramenta os abrigos poderão postar os detalhes sobre seu abrigo, animais que se encontram na disposição de ser adotado e expor necessidades que o abrigo possa estar passando. Essas necessidades podem se tratar de uma ajuda de custos a até mesmo um pedido de trabalho voluntário. As pessoas que se dispuserem a ajudar algum abrigo que está registrado na aplicação ganharão pontos que podem ser trocados por produtos relacionados ao universo pet. 3 alunos, sendo todos do sexo masculino.

Os artefatos que foram entregues em cada iteração pelas equipes são descritos na Tabela 18.

Tabela 18 – Artefatos do projeto

Iteração	**O que deve ser entregue**	**Como deve ser entregue**
0	Preenchimento do repositório público inicial do GitHub para o projeto	Página de readme/inicial com nome do projeto, justificativa do projeto, descrição do produto, especificação do papel de cada membro da equipe e demais informações pertinentes
0	Ata da reunião com o cliente e criação de histórias de usuários iniciais	Conter os participantes da reunião, tópicos discutidos e o que foi definido na mesma.As histórias devem estar no formato ("Como... Eu gostaria de... Para...)
0	Criação de mockups e storyboards Lo-Fi para histórias do usuário	
1	Criação do planejamento no formato GQM do projeto	Descrição no formato GQM na documentação visível e explícita do projeto (i.e. README do repositório)
3	Modelo ER	Representar as entidades envolvidas, seus atributos, e como eles se relacionam entre si
0 1 2 3 4	Postmorten	Resumo do que funcionou e do que não funcionou durante a iteração, e planos para melhorar o que não funcionou. a) Período: data de início - data de término da iteração b) O que estava planejado? c) O que foi feito? d) O que não foi feito? e) O que está planejado para a próxima iteração? f) Lições aprendidas (Post Mortem / Rationale)
0 1 2 3 4	Identificar o subconjunto de histórias trabalhadas na iteração	Cadastrar no github e identificar o responsável por cada uma delas e usar as labels
0 1 2 3 4	Usar BDD+TDD	Para desenvolver as histórias em Rails e implantar em Heroku
4	Um screencast curto (2-3 minutos)	Vídeo gravado pelo time mostrando um demo do projeto entregue após as 4 iterações

Fonte: Elaborado pela autora (2019)

Cada equipe contou com dois monitores(foram 12 monitores no total) que avaliavam a entrega dos artefatos conforme os critérios estabelecidos pelo professor. Na realização do projeto os alunos também fizeram uso de ferramentas CASE que foram apresentadas durante as monitorias. As ferramentas e sua utilização são apresentadas na Tabela 19.

Tabela 19 – Ferramentas CASE

Ferramenta CASE	Utilização
GitHub	Acompanhamento do projeto, entrega de artefatos e controle de versão
Cucumber e Rspec	Testes de software, garantia de qualidade e entrega de valor
Heroku	Integração, implantação e entrega

Fonte: Elaborado pela autora (2019)

6.1.3 CHECK - Analisar o progresso e desempenho

De acordo com a etapa CHECK do SELF, a avaliação do estudante deve ser guiada por instrumentos de avaliação que consideram três estratégias (avaliação de grupo/individual, conteúdo e satisfação do cliente) que devem ser consideradas durante a resolução de problemas. A Tabela 20 apresenta os tipos de avaliação para cada ciclo de aprendizagem em IF977, ao relacionar as estratégias de avaliação aos instrumentos para avaliação.

Tabela 20 – Estratégias de avaliação por ciclo de aprendizagem

Ciclo	Avaliação de grupo/individual	Conteúdo	Satisfação do cliente
0	Apresentação de Relatório; Artefatos		Apresentação de Relatório
1	Apresentação de Relatório; Artefatos		Apresentação de Relatório
2	Apresentação de Relatório; Artefatos	Prova	Apresentação de Relatório
3	Apresentação de Relatório; Artefatos		Apresentação de Relatório
4	Apresentação de Relatório; Artefatos; Entrega final; Avaliação 360°	Prova	Apresentação de Relatório; Formulário

Fonte: Elaborado pela autora (2019)

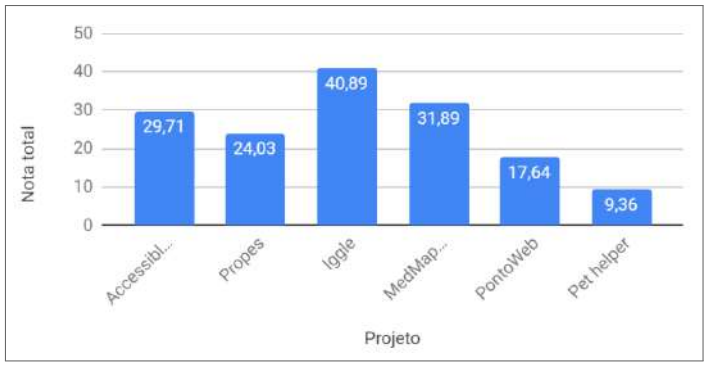
Como o ciclo 0 manteve foco no planejamento inicial do projeto, apenas as estratégias de avaliação de grupo/individual foram consideradas. O apresentação de relatório desse ciclo representa o início do projeto onde os alunos apresentam os integrantes do time e o escopo do projeto que será desenvolvido a partir da primeira reunião com o cliente. A estratégia de conteúdo não foi verificada por considerar que, no início do processo de ensino, ainda não há conteúdo suficiente para ser avaliado em forma de prova e os estudantes precisam ainda se apropriar do contexto e conceitos em ES.

Nos ciclos 1 e 2, onde ocorre o design e início de implementação do projeto, foram considerados os artefatos que se referem a essas etapas. No ciclo 2 também foi feita a avaliação da estratégia de Conteúdo, que teve como objetivo verificar a compreensão e domínio dos estudantes em relação aos requisitos de software, metodologias ágeis e resolução de problemas.

Nos ciclos 3 e 4, onde há a continuação da implementação, testes e implantação, foram considerados os artefatos correspondentes. A segunda avaliação da estratégia Conteúdo, aplicada no meio do ciclo 4, teve como objetivo verificar o entendimento dos alunos sobre técnicas e ferramentas de ES, testes de software, ciclo de vida e qualidade de software. As questões da prova abordaram os conteúdos no contexto de cada projeto. Por fim, as avaliações de entrega final; avaliação 360° e formulário de satisfação do cliente foram feitas ao fim do ciclo 4. As ordens de aplicação das estratégias de avaliação foram definidas pelo professor na etapa de planejamento.

Os resultados dessas estratégias de avaliação serão apresentados e discutidos. Em relação a estratégia Avaliação de grupo/individual, a Figura 26, apresenta o resultado dos grupos quando levado em consideração as apresentações de relatório, entrega de artefatos e entrega final e a Figura 27 apresenta o resultado das avaliações 360°.

Figura 26 – Avaliação de grupo/individual

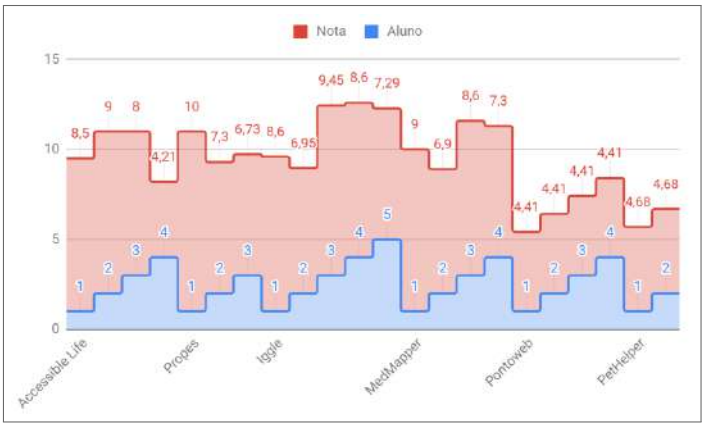


Fonte: Elaborado pela autora (2019).

A avaliação da estratégia Satisfação do cliente, levou em consideração os seguintes critérios para avaliação:

1. As reuniões serviram para elucidar questões do projeto e apresentar soluções
2. A equipe soube se portar de maneira ética nas reuniões
3. Os artefatos foram entregues nos prazos estabelecidos
4. Houve organização e planejamento da apresentação
5. A apresentação foi clara e objetiva
6. O projeto atendeu as minhas expectativas

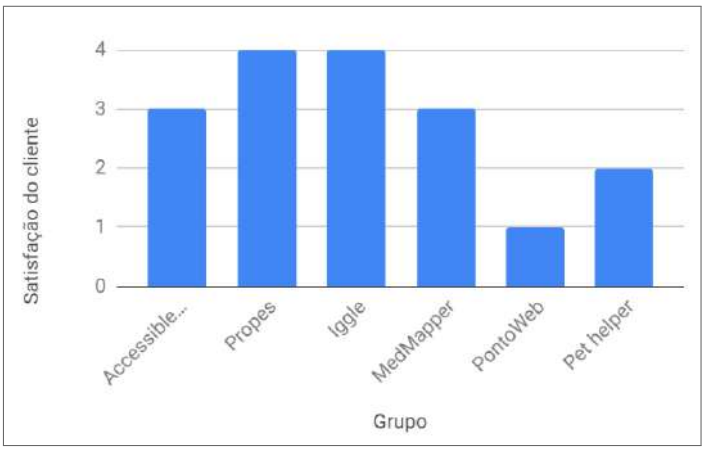
Figura 27 – Avaliações 360°



Fonte: Elaborado pela autora (2019).

É importante ressaltar que todos os times usaram os monitores e professor como clientes ao longo do processo de resolução. Cada time definiu estratégias para favorecer o contato com o cliente para, inclusive, validar os artefatos, antes do seu prazo final de entrega. A Figura 28, apresenta a média do resultado da satisfação do cliente para cada projeto.

Figura 28 – Resultado da satisfação do cliente



Fonte: Elaborado pela autora (2019).

Dentre os critérios de avaliação dos artefatos estão: prazo da entrega; se o artefato contém todas as informações necessárias e se essas informações foram apresentadas de forma clara e objetiva. Para o prazo de entrega, os alunos teriam um *deadline* final e 3 dias após essa data para que o artefato pudesse receber pontuação, ocorrendo penalidade se fosse entregue após o deadline. Supondo que o deadline de entrega de um artefato fosse dia 03/10, as equipes que entregassem o artefato até o dia 03/10 poderiam receber nota máxima de 10 pontos. Caso o artefato fosse entregue dia 04/10, a nota máxima seria 8 pontos, e caso a entrega ocorresse dia 05/10 a nota máxima seria 6 pontos. Artefatos

entregues após o dia 05/10 receberiam nota 0.

Para as apresentações de relatório, os critérios levados em consideração foram: o quanto a equipe conseguiu progredir de uma iteração para outra e o quanto cada integrante ajudou neste progresso ou não, visto que em cada apresentação era apresentado tudo que foi realizado durante a iteração e por quem. Durante a apresentação o professor formulava perguntas para instigar o grau de aprendizado nos alunos em determinado conteúdo, como por exemplo "Qual a técnica de teste utilizada no projeto? Por que não utilizaram outra técnica?".

A entrega final leva em consideração as apresentações de todas as iterações, a satisfação do cliente e a execução do projeto, levando em consideração como os alunos organizaram e preparam as atividades, como atuaram no gerenciamento de problemas, etc.

Na avaliação 360° os integrantes da equipe avaliavam uns aos outros levando em consideração a proatividade, organização, disponibilidade e forma de trabalhar em equipe de cada um. Com isso, foi decidido qual a nota individual de cada integrante, tomando como base a nota da Entrega final da equipe. Cada equipe teve uma nota máxima de entrega final baseada na quantidade de alunos por equipe, logo, se uma equipe possuía 5 integrantes, sua nota máxima seria 50. Na avaliação 360° os alunos precisaram distribuir esta nota pela quantidade de integrantes.

Logo, a nota final do projeto ficou dividida desta maneira: **Nota Final do Projeto (PRJ) = ([EQP] + [EXT] + [PEN])**, onde

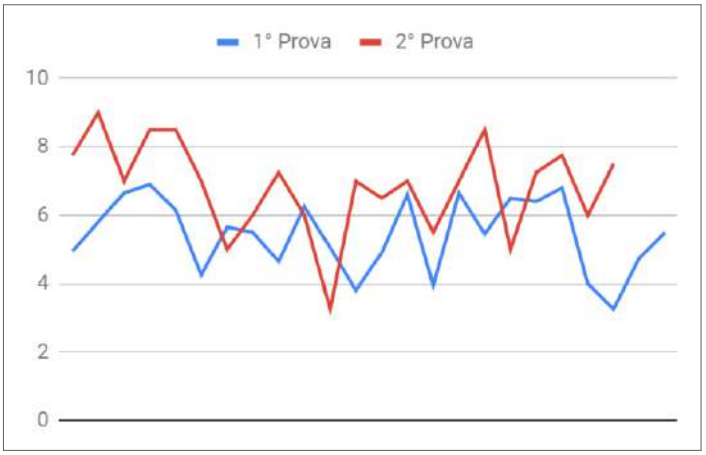
- EQP: nota definida pela equipe a partir da distribuição da nota do projeto (Avaliação 360°);
- EXT: pontos extras atribuídos a partir da execução dos exercícios práticos (Monitorias);
- PEN: penalizações diversas por não comparecimento às apresentações ou a avaliações específicas do Projeto (valor negativo).

Em relação a estratégia conteúdo, a média da turma para a primeira avaliação foi de 5,43. Para a primeira prova aplicada, nenhum dos estudantes, obtiveram desempenho igual ou superior à média desejada (7,0). Em comparação a estratégia de avaliação em grupo/individual, este resultado indica que houve dificuldade dos alunos em demonstrar compreensão pela informação,não sendo capaz de reproduzir a mesma por ideias e palavras próprias, visto que as questões da prova foram, em seu total, questões discursivas.

Para a segunda prova, os resultados foram melhores, com a média da turma subindo para 6,83 e 56% dos estudantes com desempenho igual ou superior à média desejada (7,0). Isto indica que, a partir de situações de aprendizagem anteriores, os estudantes puderam fundamentar a compreensão sobre os assuntos abordados e perceberam a importância de se colocarem como autores do próprio aprendizado, não esperando apenas que o professor

oferecesse uma lista de soluções para um determinado problema. A Figura 29, apresenta o resultado do desempenho individual dos alunos da turma para as duas provas aplicadas. Dois alunos não participaram da 2° prova.

Figura 29 – Resultado de desempenho individual



Fonte: Elaborado pela autora (2019).

A nota final de cada aluno foi calculada da seguinte forma: **Média = (3 * Prova1 + 3 * Prova2 + 4 * Projeto)/ 10**. Estes resultados demonstram uma avaliação tendenciosamente positiva em termos de participação dos alunos na realização do projeto visto que apenas dois grupos obtiveram todas as notas abaixo da média esperada (7,0). É importante observar que o nível de participação está diretamente relacionado ao nível de aprendizagem, visto que o critério de avaliação levou em consideração o poder de argumentação dos alunos com foco no conteúdo estudado em cada aula e desenvolvido durante o projeto.

O modelo de avaliação através de provas, buscou desenvolver conhecimento nos alunos através da capacidade de lembrar do conteúdo discutido em sala de aula, para que os mesmos pudessem sugerir soluções de software para sistemas de informação.

O conhecimento prévio dos alunos, adquirido durante as aulas expositivas pôde ser transferido para a realização dos exercícios práticos o que permitiu o professor avaliar que tais experiências de aprendizagem permitiram dar aos alunos um nível de compreensão que os permitiria produzir software para o atendimento de necessidades sociais ou organizacionais.

Em seguida os alunos puderam aplicar o conhecimento adquirido nas aulas e nas monitorias durante a execução do projeto prático. Dessa forma, os alunos foram estimulados a aplicar processos, técnicas e ferramentas de desenvolvimento de software, além de ter a oportunidade de desenvolver o compartilhamento de conhecimento, bem como de comunicação, negociação, colaboração e liderança.

Evidentemente que é difícil analisar e definir o grau de subjetividade e presteza do professor neste processo avaliativo, principalmente considerando o elevado número de

alunos(25 alunos) envolvidos no processo. No entanto, esta ferramenta traz, independentemente deste aspecto, um valor intrínseco, visto que forçou o professor a estimular e estar atento à participação de todos os alunos e a estes de procurar melhorar suas performances, visto que eles obtiveram *feedback* do professor ao final de cada apresentação.

6.1.4 ACT - Avaliar a execução do processo

De acordo com a etapa ACT do SELF, a qualidade do processo de ensino e aprendizagem é verificada através de questionário aplicado aos alunos. O objetivo do questionário é coletar dados sobre a avaliação do ensino por parte dos alunos. Os resultados do questionário são peças fundamentais que fornecem subsídios para desenvolvimento desta fase do ciclo que visa melhorar o processo de ensino-aprendizagem.

Os alunos responderam perguntas sobre pontos fortes e fracos relacionadas ao método de ensino e disciplina no geral; como também aspectos gerais sobre a disciplina e o professor. Também puderam fazer sugestões para melhorar a qualidade da disciplina. Nas subseções a seguir encontram-se os resultados dos questionários.

6.1.4.1 Pontos fortes e/ou fracos de ensino do professor

> "As aulas são bastante didáticas e divertidas, a abordagem do professor é muito convidativa a participação."

> "Gostei bastante do professor, sempre bastante atencioso e disponível. Tirou minhas dúvidas rapidamente e de forma clara"

> "Bom, as aulas expositivas serviam mais para dar uma visão geral da temática. Mas para ser aplicado no projeto você necessariamente precisa se aprofundar mais. Acho que as aulas expositivas poderiam ter sido mais resumidas em tópicos e direcionadas para conteúdo externo."

6.1.4.2 Pontos fortes e/ou fracos da disciplina em geral

> "O projeto ajudou a edificar vários conhecimentos, não só de desenvolvimento, mas também de organização. "

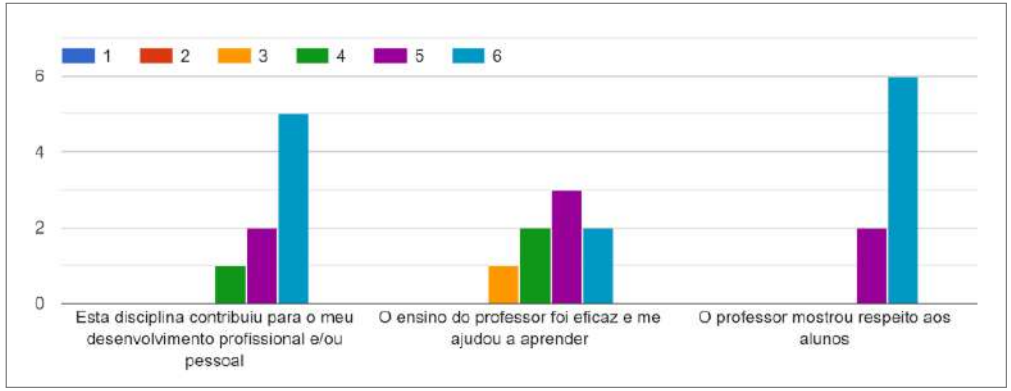
> "Os feedbacks do projeto demoraram bastante e isso nos prejudicou pois não sabíamos se estávamos no caminho correto"

> "As entregas poderiam ser melhor documentadas, algumas vezes o syllabus dizia uma coisa, a planilha de entregas da página do projeto dizia outra"

6.1.4.3 Declarações gerais sobre a disciplina e o professor

1- Discordo muito fortemente 2-Discordo fortemente 3-Discordo 4-Concordo 5-Concordo fortemente 6-Concordo muito fortemente

Figura 30 – Declarações gerais sobre a disciplina e o professor



Fonte: Elaborado pela autora (2019).

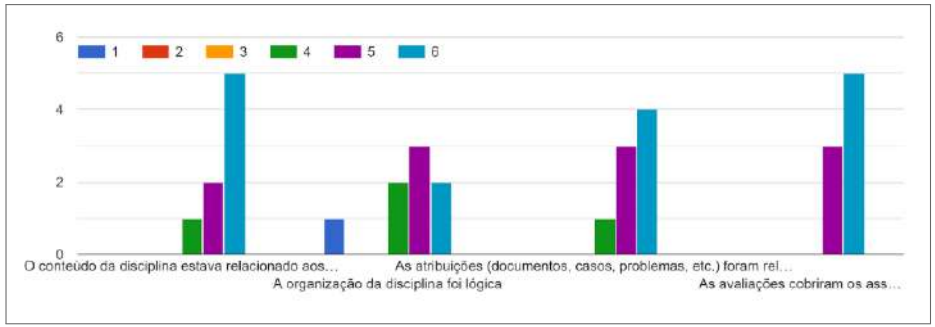
6.1.4.4 Aspectos específicos da disciplina

1. O conteúdo da disciplina estava relacionado aos objetivos de aprendizagem
2. A organização da disciplina foi lógica
3. As atribuições (documentos, casos, problemas, etc.) foram relacionadas ao conteúdo da disciplina
4. As avaliações cobriram os assuntos abordados na disciplina

6.1.4.5 Outros aspectos da disciplina

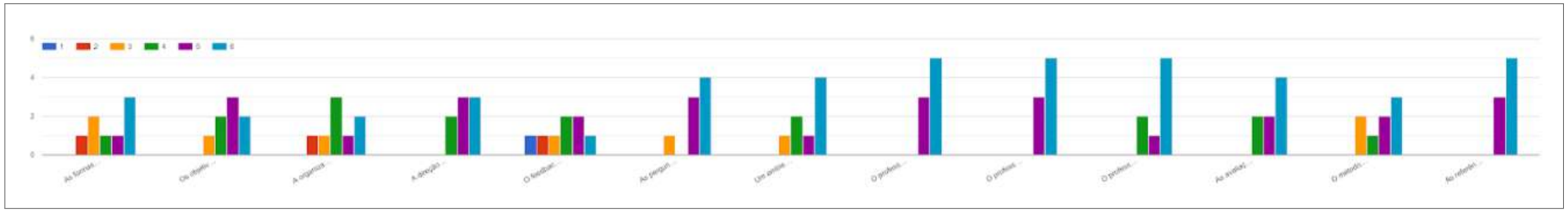
1-Nunca 2-Raramente 3-Ocasionalmente 4-Geralmente 5-Frequentemente 6-Sempre

Figura 31 – Aspectos específicos da disciplina



Fonte: Elaborado pela autora (2019).

Figura 32 – Outros aspectos da disciplina



Fonte: Elaborado pela autora (2019).

1. As formas de avaliação da disciplina foram claramente explicados
2. Os objetivos de aprendizagem da disciplina foram claramente declarados e explicados
3. A organização da disciplina (syllabus) e suas expectativas foram claramente declarados e explicados
4. A direção da disciplina e as formas de avaliação foram compreensíveis
5. O feedback das avaliações foram oportunos e adequados
6. As perguntas feitas pelos alunos foram claramente e adequadamente respondidas
7. Um ambiente de aprendizado produtivo foi mantido em cada aula
8. O professor estava disponível para consulta
9. O professor pareceu estar bem preparado para cada aula
10. O professor falou claramente o suficiente para ser entendido
11. As avaliações foram utilizados para auxiliar na aprendizagem
12. O método de ensino utilizado me ajudou a aprender
13. As referências, livros e/ou apostilas utilizadas me ajudaram a aprendizagem

Os alunos participantes do questionário também destacaram que a realização das atividades práticas juntamente com a exposição de conteúdos teóricos mudou a percepção sobre a área de TI, dando uma ideia mais realista do que é a área de desenvolvimento, projetos e engenharia de software que, até então, eles não conseguiam enxergar como o conhecimento era aplicado no mercado e rotina de empresas de TI.

Além disso, os alunos relataram que o uso de estudos independentes foram de grande importância, pois através da necessidade, costumavam seguir atrás de novos conhecimentos que os fizeram crescer intelectual e profissionalmente. Alguns alunos relataram que a falta de comprometimento de alguns membros da equipe do projeto, fizeram com que as entregas do mesmo fossem afetadas em questão de prazo e qualidade, porém até isso os ensinou lições valiosas no que diz respeito a de gestão de pessoas.

A aproximação dos estudantes ao seu próprio aprendizado, tal como foi viabilizado na disciplina, reflete a contribuição para o desenvolvimento de aprendizagens plurais, significativas, como também ajuda o aluno a ter uma visão mais crítica do processo de ensino-aprendizagem, visto que este não está mais centrado no protagonismo do professor e na passividade do estudante. Assim, quanto mais rápido o professor perceber a importância do diálogo como um aspecto necessário em suas aulas, maiores avanços estará conquistando em relação aos alunos, pois desse modo, poderá despertar a curiosidade dos mesmos e os mobilizará a transformarem a realidade.

A partir desses resultados, percebeu-se que os aspectos que mais impactaram negativamente o processo de ensino e aprendizagem estão relacionados a demora de *feedback* e a organização da disciplina quanto ao agrupamento de informações. Recentemente o site da disciplina foi realocado para um repositório no *gitHub*, o que fez com houvesse informações desencontradas nas páginas, pois algumas abas do novo repositório acabaram ficando com informações passadas. Também houve a entrada de vários monitores novos que precisaram se adequar ao ritmo de correções e *feedbacks*, o que fez que com os *feedbacks* levassem mais tempo que o esperado, deixando assim os alunos sem saber se os artefatos entregues estavam de acordo com o previsto.

Para mitigar esses desvios na próxima rodada da disciplina, será feita uma revisão em todo repositório atual da disciplina para que as informações estejam agrupadas e corretas em um único lugar. Também haverá uma reunião com os monitores antes do semestre letivo para que os mesmos estejam cientes dos prazos de *feedback* e da importância que eles sejam feitos dentro deste prazo. Ao início da disciplina o professor também deixará mais evidente como ocorrerá as avaliações da disciplina e que os estudantes precisarão se aprofundar em certos conteúdos (que podem ou não ser indicados pelo professor) de forma proativa fora do ambiente de sala de aula.

6.2 AVALIAÇÃO DO SELF

A avaliação da documentação do *framework* foi realizada através de um painel de especialistas, um método empírico baseado na opinião de especialistas (BEECHAM et al., 2005). O critério para seleção dos especialistas foi: ter experiência docente na área de ES.

As perguntas de pesquisa foram derivadas do mapeamento sistemático de (Marques; Quispe; Ochoa, 2014), que destaca que os professores consideram diversos critérios na seleção de abordagens de ensino, como: a flexibilidade e a facilidade na utilização das abordagens; a adequação da abordagem e o esforço necessário para serem utilizadas pela maioria dos instrutores; as restrições e as competências envolvidas na utilização dessas abordagens. Assim, foi elaborado um questionário auto administrado com 17 perguntas que buscam avaliar a efetividade da especificação do *framework* como mostra a Tabela 21.

Tabela 21 – Perguntas da Avaliação por Painel de Especialistas

Categoria	**Critério**	**Pergunta**
Documentação	Adequação	O texto da documentação está adequado para o tema abordado pelo framework(Ensino de Engenharia de Software)?
	Clareza	O texto da documentação está claro e objetivo?
	Completude	O texto da documentação apresenta os componentes necessários à especificação de um framework de ensino?
	Consistência	Os componentes estão consistentes com o foco do framework de ensino?
	Corretude	A documentação descreve os componentes do framework de maneira correta? Não há erros de sintaxe ou semântica?
	Detalhamento	O nível de detalhes é suficiente para fornecer uma base adequada para o uso do framework de ensino?
Usabilidade	Facilidade de uso	O framework é fácil de usar?
	Flexibilidade de uso	O framework permite flexibilidade no seu uso?
	Adequação	O framework é adequado para ser usado pela maioria dos professores?
	Esforço envolvido	Qual o grau de esforço envolvido para um professor usar o framework?
	Habilidades requeridas	Qual o grau de habilidades requeridas para um professor usar o framework?
	Restrições de uso	Qual o grau de restrições para usar o framework?
	Utilidade	O quão útil você considera o framework apresentado no apoio ao ensino de Engenharia de Software?
	Recomendação	O quanto você recomendaria esse framework para uso em sala de aula no ensino de Engenharia de Software?

Fonte: Elaborado pela autora (2019)

Foram realizadas 14 perguntas objetivas, sendo 6 sobre a documentação, 6 sobre a usabilidade e 2 para emissão do parecer final. Nas opções de respostas para as questões sobre a documentação, o especialista poderia responder sim ou não e emitir uma consideração subjetiva sobre a resposta. Já nas opções de respostas para as questões sobre a usabilidade e parecer final foi utilizada uma escala Likert de 0 a 5 pontos. Além destas, realizaram-se 3 perguntas subjetivas no parecer final sobre os pontos fortes, pontos fracos e considerações finais. O processo de avaliação ocorreu através de uma análise e avaliação individual do *framework* (via questionário no *Google Forms*) no período de 01 de outubro de 2019 a 09 de outubro de 2019.

6.2.1 Participantes

Foram identificados e convidados, via e-mail, 3 (três) especialistas, com média de experiência docente de 8 anos. Esses professores são de universidades públicas, sendo a professora Ana Paula Furtado da Universidade Federal Rural de Pernambuco (UFRPE), o professor Felipe Silva Ferraz do Centro de Estudos e Sistemas Avançados do Recife (CESAR School) e o professor Flávio da Silva Neves do Instituto Federal de Pernambuco (IFPE). Quanto à experiência, todos os especialistas já realizaram projetos práticos, discussão de casos práticos e adotaram PBL em sala de aula.

6.2.2 Resultados

A partir da análise das respostas do questionário, como mostra a Tabela 22, conclui-se que o painel de especialistas considerou o *framework* completo e consistente. Os professores também destacaram que o *framework* fornece uma base adequada para seu uso no ensino de ES.

Além das respostas mostradas na Tabela 22, os especialistas também informaram que os pontos fortes do *framework* são:

> Organização e apoio ao professor para montar um curso.
>
> O FM ajuda organizar a forma de planejar a disciplina, facilita sua organização
>
> O principal ponto forte do *framework* é que ao mesmo tempo que ele oferece uma padronização para condução da disciplina, também prevê maneira de se adaptar a possíveis alterações no decorrer da disciplina.

E como pontos fracos, indicaram que é necessário aplicar o mesmo com turmas reais, diante de algumas demandas que os professores tem no seu cotidiano.

Também foi levantada, por um dos especialistas a questão de que para professores que não adotem metodologias ativas, por exemplo, pode haver dificuldade na compreensão de alguns aspectos do *framework*. Para isso, foi inserida uma subseção na documentação a

Tabela 22 – Respostas da Avaliação por Painel de Especialistas

Categoria	**Critério**	**Avaliador 1**	**Avaliador 2**	**Avaliador 3**
Documentação	Adequação	SIM	SIM	SIM
	Clareza	SIM	SIM	SIM
	Completude	SIM	SIM	SIM
	Consistência	SIM	SIM	SIM
	Corretude	NÃO	SIM	NÃO
	Detalhamento	SIM	SIM	SIM
Usabilidade	Facilidade de uso	Não consigo avaliar	Concordo	Concordo
	Flexibilidade de uso	Concordo Totalmente	Concordo	Concordo
	Adequação	Concordo Parcialmente	Concordo Totalmente	Concordo
	Esforço envolvido	Muito Esforço	Muito Esforço	Esforço Moderado
	Habilidades requeridas	Não consigo avaliar	Habilidades Básicas	Habilidades Moderadas
	Restrições de uso	Restrições Básicas	Poucas Restrições	Poucas Restrições
	Utilidade	Muito útil	Moderadamente útil	Muito útil
	Recomendação	Recomendaria Largamente	Recomendaria Moderadamente	Recomendaria Moderadamente

Fonte: Elaborado pela autora (2019)

fim de especificar o perfil do professor que é público-alvo do *framework* e assim estabelecer as habilidades mínimas que serão requeridas.

6.3 CONCLUSÕES DO CAPÍTULO

Este capítulo apresentou os resultados da aplicação e avaliação do *framework* proposto. Inicialmente, o *framework* foi aplicado na disciplina de Engenharia de Software do 4° período do curso de Sistemas de Informação e foi apresentado como o SELF ajudou o professor a desenvolver a disciplina de forma mais objetiva como também a descobrir problemas inerentes ao processo de ensino-aprendizagem de forma mais precisa. Apesar de ter sido uma aplicação única e a seu contexto, o *framework* foi desenvolvido para ser aplicado em qualquer tipo de disciplina de ES. Provavelmente, o que deve mudar é a forma que será aplicado (levando em consideração o arranjo produtivo local e os objetivos particulares do professor para com a disciplina) e não as etapas de aplicação.

Posteriormente, o *framework* foi avaliado por especialistas quanto a sua usabilidade e documentação. Entre os resultados da aplicação e avaliação do framework destacam-se:

- O gerenciamento do processo de ensino-aprendizagem segundo os princípios do PDCA e o investimento permanente no desenvolvimento dos estudantes, colocando-os como protagonistas deste processo, são a base para uma educação transformadora em níveis educacionais superiores.

- O framework pode facilitar e padronizar a execução de todo o processo de ensino-aprendizagem, através do gerenciamento da carga horária disponível e adequando o ensino dos conteúdos selecionados.

- A adoção do framework permite que os alunos executem atividades práticas de maneira iterativa através de ciclos de aprendizagem, assim, ao avançar em cada etapa do ciclo, o aluno poderá conhecer, compreender e aplicar os conteúdos da ES e, consequentemente, desenvolver as competências desejadas.

7 CONCLUSÃO

O objetivo deste capítulo é apresentar as considerações finais e contribuições decorrentes desta pesquisa assim como seus trabalhos futuros associados.

7.1 CONSIDERAÇÕES FINAIS E CONTRIBUIÇÕES

Esta pesquisa de mestrado identificou três lacunas na área de ensino de ES: i) o ensino em sala de aula não é efetivo; ii) restrições de tempo; e iii) falta de habilidades práticas nos alunos. Assim, a principal Questão de Pesquisa (QP) investigada por esta tese foi: "Como professores da área de Engenharia de Software podem construir uma disciplina que desenvolva competências técnicas e práticas nos alunos?".

Nesse sentido, o objetivo desta pesquisa consistiu na definição de um *framework* baseado no ciclo PDCA para apoiar o ensino de Engenharia de Software. Como diferencial, esse *framework* buscou auxiliar o professor na construção de uma disciplina do início ao fim para que os estudantes desenvolvessem competências técnicas e trabalhassem a parte prática em ES.

Buscando entender o problema, inicialmente realizou-se um *survey* com professores, alunos e profissionais da área de ES, onde realizaram-se perguntas descritivas e classificatórias relacionadas aos tópicos, abordagens e principais competências em ES.

A maioria dos professores respondeu que adota abordagens focadas no professor, como aulas expositivas, reforçando as problemáticas identificadas na literatura relacionadas à lacuna (i). Por outro lado, destaca-se que a grande maioria dos alunos entrevistados no *survey* considera efetivo para o ensino de ES a realização de projetos de software e atividades práticas, pois estas contribuem para construir uma base prática fundamentada em conceitos teóricos comprovados. Para os profissionais, a principal competência que um engenheiro de software precisa ter é "Identificar, especificar e validar os requisitos, projetar, implementar, verificar, implantar e documentar soluções de software baseadas no conhecimento apropriado de teorias, modelos e técnicas."

Quanto aos tópicos de ES, correlacionou-se o percentual de tópicos considerados relevantes pelos professores com o percentual de aprendizagem dos alunos. Assim, identificaram-se as 5 (cinco) unidades de conhecimento mais adotadas no ensino de ES: Engenharia de Requisitos, Teste de Software, Processos de ES, Construção de Software e Design de Software.

Além da questão de pesquisa, com o intuito de refinar a investigação, considerou-se quatro questões gerais. Em resposta a **Q1 (Quais os desafios que um professor pode se deparar ao ensinar uma disciplina de ES?)** a pesquisa observou que alguns desafios são comuns às pesquisas que relatam sobre o ensino de ES. No capítulo 2 é

destacado evidências que refletem essas dificuldades: carga horária insuficiente para cobrir todos os tópicos de ES em uma única disciplina, dificuldade em trazer experiências práticas válidas para a sala de aula e falta de interesse dos alunos por conteúdos puramente teóricos.

Em resposta a **Q2 (Quais soluções existentes podem contribuir para o ensino de ES, de forma a reduzir os impactos dos desafios?)** a pesquisa identificou que a maioria das soluções evidenciam o que o professor pode fazer para melhorar o processo de ensino-aprendizagem, mas não focam em como ele pode construir a disciplina do início ao fim para trabalhar habilidades específicas nos alunos. A Seção 3.5.1 apresenta soluções como: *framework* HOT que atenta para a necessidade do ensino não abordar apenas os aspectos tecnológicos da engenharia de software, mas em vez disso, ele deve referir também ao ambiente de trabalho e ao quadro profissional (HAZZAN; DUBINSKY, 2010); FASENG, *framework* para jogos de simulação de Engenharia de Software (PEIXOTO et al., 2012); FRAMES, um *framework* que foi estruturado de forma a ser aplicado a qualquer unidade de conhecimento da Engenharia de Software (PORTELA; VASCONCELOS; OLIVEIRA, 2016).

Em resposta a **Q3 (Quais as características desejáveis de uma solução que dê suporte ao ensino de competências técnicas e habilidades práticas em ES?)** a pesquisa considera que investir na construção de uma disciplina desde seu início para trabalhar as competências desejadas nos alunos é o segredo para alcançar os objetivos esperados na implementação do ensino de ES. Portanto, estabelecer esta relação em uma solução significou considerar o aprendizado que os alunos obtiveram em disciplinas anteriores, como também características que referenciam modelos, técnicas e gestão de processos de ensino-aprendizagem que tiram o foco do professor e colocam o aluno como centro.

No SELF, estas características são evidenciadas em sua estrutura e componentes destacados no capítulo 5, ao considerar: 1) kaizen como filosofia para definição de padrões e melhoria contínua; 2) fases baseadas no ciclo PDCA para identificar problemas, analisá-los, propor e executar novos procedimentos; 3) ênfase na abordagem humanista, pois ela possui as características como foco no aluno e vê o professor como um facilitador da aprendizagem criando condições para que os alunos aprendam; e 4) uso de características principais do PBL como componente para desenvolver competências nos alunos e conduzir a dinâmica da execução do processo de ensino-aprendizado.

Esta proposta favorece uma forma de ensino que tem como objetivo produzir competências mensuráveis em situações de atividades específicas, e para isso, recomenda-se o uso de metodologias que buscam aproximar a teoria da prática. Quanto a avaliação destas competências, a recomendação é que seja focada no que o aluno é capaz de fazer, dando ênfase a capacidade de aplicação e síntese do conhecimento, e não a sua obtenção propriamente dita. Por fim, para o professor, espera-se uma mudança da sua relação com o aluno e o processo de ensino-aprendizagem, já que seu propósito principal agora é de fazer com que o aluno aprenda e não apenas transmitir informações sobre certos conteúdos.

O foco em competências também pretende responder às novas exigências da indústria,

realizando um esforço de maior proximidade com as demandas e objetivando estar em conformidade com a parcialidade dos alunos, tendo em vista responder às novas características do mercado.

7.2 TRABALHOS FUTUROS

Por ter apenas uma aplicação em que o SELF é testado, o trabalho limita-se ao contexto desta aplicação: uma Instituição Federal de Ensino Superior. Como trabalho futuro, destacam-se:

- Posteriormente, pretende-se realizar um Estudo de Caso que permita observar como ocorre o uso do SELF pelos professores e se tal uso tem relação direta com o desenvolvimento das competências esperadas nos alunos de ES.
- Aplicar e avaliar o SELF em disciplinas de ES de outras instituições de ensino.

Academicamente, pretende-se escrever e publicar artigos sobre a aplicação e avaliação do SELF através do estudo de caso. O objetivo é corroborar os resultados aqui apresentados e participar de eventos e conferências a fim de discutir com pesquisadores da área sobre os resultados desta tese. Dessa forma, buscar-se-á manter o *framework* atualizado e adequado para o ensino de ES e, consequentemente, eficiente para o desenvolvimento de uma disciplina que trabalhe competências técnicas e habilidades práticas nos alunos.

REFERÊNCIAS

ALEXANDRE, G.; SANTOS, S. Poster: Pbl planner toolkit: A canvas-based tool for planning pbl in software engineering education. In: IEEE. *2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion)*. [S.l.], 2018. p. 153–154.

AMADOR, J. A.; MILES, L.; PETERS, C. B. The practice of problem based learning. 2007.

ANTUNES, C. *Como desenvolver as competências em sala de aula*. [S.l.]: Editora Vozes Limitada, 2011.

BAKER, A.; NAVARRO, E. O.; HOEK, A. V. D. An experimental card game for teaching software engineering processes. *Journal of Systems and Software*, Elsevier, v. 75, n. 1-2, p. 3–16, 2005.

BARNES, R. M. *Estudo de movimentos e de tempos: projeto e medida do trabalho*. [S.l.]: Editora Edgard Blu cher, 1977.

BECK, K.; GAMMA, E. *Extreme programming explained: embrace change*. [S.l.]: addison-wesley professional, 2000.

BECKMAN, K.; ALONIS, J. Directory of industry and university collaborations with a focus on software engineering education and training. Citeseer, 1997.

BEECHAM, S.; HALL, T.; BRITTON, C.; COTTEE, M.; RAINER, A. Using an expert panel to validate a requirements process improvement model. *Journal of Systems and Software*, Elsevier, v. 76, n. 3, p. 251–275, 2005.

BENTON, S. L.; CASHIN, W. E.; KANSAS, E. Idea paper# 50 student ratings of teaching: A summary of research and literature. Citeseer, 2012.

BLOOM, B. S. et al. Taxonomy of educational objectives. vol. 1: Cognitive domain. *New York: McKay*, p. 20–24, 1956.

BORDENAVE, J. E. D.; PEREIRA, A. M. Estratégias de ensino-aprendizagem. In: *Estrategias de ensino-aprendizagem*. [S.l.]: Vozes, 1985.

BOTERF, G. L. et al. Les éditions d'organisations. *Paris: Quatrième Tirage*, 1995.

BOURQUE, P.; FAIRLEY, R. E. et al. *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. [S.l.]: IEEE Computer Society Press, 2014.

BOURQUE, P.; ROBERT, F. Curriculum guidelines for undergraduate degree programs in software engineering. *IEEE Computer Society*.

BRITO, M. S.; SILVA, F. G.; CHAVEZ, C. v. F. G.; NASCIMENTO, D. C.; BITTENCOURT, R. A. Floss in software engineering education: an update of a systematic mapping study. In: ACM. *Proceedings of the XXXII Brazilian Symposium on Software Engineering*. [S.l.], 2018. p. 250–259.

CORDEIRO[1], A. M.; OLIVEIRA, G. M. de; RENTERÍA-TCBC-RJ, J. M.; GUIMARÃES-TCBC-RJ, C. A. Revisão sistemática: uma revisão narrativa. SciELO Brasil, 2007.

CURRICULA, A. f. C. M. A. Joint Task Force on C.; SOCIETY, I. C. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA: ACM, 2013. 999133. ISBN 978-1-4503-2309-3.

CURRICULA, T. J. T. F. on C. *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. New York, NY, USA, 2004.

CURRICULA, T. J. T. F. on C. *Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA, 2008.

DING, J. A framework for global collaboration in teaching software engineering. In: IEEE. *2013 3rd International Workshop on Collaborative Teaching of Globally Distributed Software Development (CTGDSD)*. [S.l.], 2013. p. 30–34.

DUCH, B. J.; GROH, S. E.; ALLEN, D. E. Why problem-based learning? a case study of institutional change in undergraduate education. *The power of problem-based learning*, v. 4, 2001.

ELIAS, C. d. S. R.; SILVA, L. A. da; MARTINS, M. T. d. S. L.; RAMOS, N. A. P.; SOUZA, M. d. G. G. de; HIPÓLITO, R. L. Quando chega o fim? uma revisão narrativa sobre terminalidade do período escolar para alunos deficientes mentais. *SMAD, Revista Electrónica en Salud Mental, Alcohol y Drogas*, Universidade de São Paulo, v. 8, n. 1, p. 48–53, 2012.

ERICONE, D. *Ser professor*. [S.l.]: Edipucrs, 2001.

ERL, T. *Service-oriented architecture: analysis and design for services and microservices*. [S.l.]: Prentice Hall Press, 2016.

FABBRI, S. C. P. F.; FERRARI, F. C.; CAMARGO, K. G. A atividade de teste sob a perspectiva de qualidade de software. *Revista TIS*, v. 2, n. 3, 2014.

FILHO, J. L. *Projeto conceitual de banco de dados geográficos através da reutilização de esquemas, utilizando padrões de análise e um framework conceitual*. Tese (Doutorado) — Instituto de Informática da Universidade Federal do Rio Grande do Sul., 2000.

FLEURY, M. T. L.; FLEURY, A. Construindo o conceito de competência. *Revista de administração contemporânea*, SciELO Brasil, v. 5, n. SPE, p. 183–196, 2001.

FOX, A.; PATTERSON, D. Crossing the software education chasm. *Communications of the ACM*, ACM, v. 55, n. 5, p. 44–49, 2012.

GARG, K.; VARMA, V. Case studies as assessment tools in software engineering classrooms. In: IEEE. *2009 22nd Conference on Software Engineering Education and Training*. [S.l.], 2009. p. 8–11.

Ghezzi, C.; Mandrioli, D. The challenges of software engineering education. In: *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.* [S.l.: s.n.], 2005. p. 637–638. ISSN 0270-5257.

GIBBS, R. W.; JR, R. W. G.; GIBBS, J. *The poetics of mind: Figurative thought, language, and understanding*. [S.l.]: Cambridge University Press, 1994.

GIL, A. C. Como elaborar projetos de pesquisa. são paulo: Atlas, 2006. gil, antônio carlos. *Como elaborar projetos de pesquisa*, v. 5, 2010.

GIMENES, I. Os dilemas didáticos da engenharia de software. *Revista da Sociedade Brasileira de ComputaçãoBrasil, Edição*, v. 28, 2015.

GNATZ, M.; KOF, L.; PRILMEIER, F.; SEIFERT, T. A practical approach of teaching software engineering. In: IEEE. *Proceedings 16th Conference on Software Engineering Education and Training, 2003.(CSEE&T 2003)*. [S.l.], 2003. p. 120–128.

GOMES, L. *5W2H: Ferramenta para a elaboração de Planos de Ação*. [S.l.]: Recuperado, 2014.

GONZÁLEZ-MORALES, D.; ANTONIO, L. M. M. D.; GARCÍA, J. L. R. Teaching "soft" skills in software engineering. In: IEEE. *2011 IEEE Global Engineering Education Conference (EDUCON)*. [S.l.], 2011. p. 630–637.

GRILLO, M. C. Práticas docentes e referenciais norteadores. *Caderno Marista de Educação*, p. 41–52, 2002.

HAZZAN, O.; DUBINSKY, Y. Teaching a software development methodology: the case of extreme programming. In: IEEE. *Proceedings 16th Conference on Software Engineering Education and Training, 2003.(CSEE&T 2003)*. [S.l.], 2003. p. 176–184.

HAZZAN, O.; DUBINSKY, Y. Teaching framework for software development methods. In: *Proceedings of the 28th International Conference on Software Engineering*. New York, NY, USA: ACM, 2006. (ICSE '06), p. 703–706. ISBN 1-59593-375-1. Disponível em: <http://doi.acm.org/10.1145/1134285.1134396>.

HAZZAN, O.; DUBINSKY, Y. A hot—human, organizational and technological—framework for a software engineering course. In: IEEE. *2010 ACM/IEEE 32nd International Conference on Software Engineering*. [S.l.], 2010. v. 1, p. 559–566.

HERBERT, N.; SALAS, K. de; LEWIS, I.; DERMOUDY, J.; ELLIS, L. Ict curriculum and course structure: the great balancing act. In: AUSTRALIAN COMPUTER SOCIETY, INC. *Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148*. [S.l.], 2014. p. 21–30.

HESSE-BIBER, S. N. *Mixed methods research: Merging theory with practice*. [S.l.]: Guilford Press, 2010.

HEVNER, A.; MARCH, S. T.; PARK, J.; RAM, S. Design science research in information systems. *MIS quarterly*, v. 28, n. 1, p. 75–105, 2004.

HILBURN, T. B.; TOWHIDNEJAD, M. A case for software engineering. In: IEEE. *20th Conference on Software Engineering Education & Training (CSEET'07)*. [S.l.], 2007. p. 107–114.

HUNG, W.; JONASSEN, D. H.; LIU, R. et al. Problem-based learning. *Handbook of research on educational communications and technology*, Citeseer, v. 3, n. 1, p. 485–506, 2008.

HUNT, V. D. *Process mapping: how to reengineer your business processes*. [S.l.]: John Wiley & Sons, 1996.

HWANG, D.; MA, Z.; WANG, M. The information systems core: A study from the perspective of is core curricula in the us. *Information Systems Education Journal*, v. 13, n. 6, p. 27, 2015.

IBRAHIM, N.; HALIM, S. A. Generic framework design of project-oriented problem-based learning (popbl) for software engineering courses. In: IEEE. *2014 8th. Malaysian Software Engineering Conference (MySEC)*. [S.l.], 2014. p. 359–364.

IMAI, M. *Kaizen: a estratégia para o sucesso competitivo*. [S.l.]: Imam, 1994.

JABAREEN, Y. Building a conceptual framework: philosophy, definitions, and procedure. *International journal of qualitative methods*, SAGE Publications Sage CA: Los Angeles, CA, v. 8, n. 4, p. 49–62, 2009.

JUNIOR, C. D. M.; TANI, G.; MANOEL, E. D. J. A estrutura da prática variada em situações reais de ensino-aprendizagem. *Revista Brasileira de Ciência e Movimento*, v. 9, n. 4, p. 55–64, 2008.

KAJKO-MATTSSON, M.; SJÖGREN, A.; LINDBÄCK, L. Everything is possible to structure-even the software engineering body of knowledge. In: IEEE. *2017 IEEE/ACM 1st International Workshop on Software Engineering Curricula for Millennials (SECM)*. [S.l.], 2017. p. 61–67.

KASANEN, E.; LUKKA, K.; SIITONEN, A. The constructive approach in management accounting research. *Journal of management accounting research*, Sarasota, v. 5, n. 1, p. 243–264, 1993.

KITCHENHAM, B.; BUDGEN, D.; BRERETON, P.; WOODALL, P. An investigation of software engineering curricula. *Journal of Systems and Software*, Elsevier, v. 74, n. 3, p. 325–335, 2005.

KITCHENHAM, B. A.; PFLEEGER, S. L. Personal opinion surveys. In: *Guide to advanced empirical software engineering*. [S.l.]: Springer, 2008. p. 63–92.

KLINK, M. Van der; BOON, J.; SCHLUSMANS, K. Competências e ensino superior profissional: presente e futuro. *Revista Europeia de Formação Profissional*, v. 40, n. 1, p. 72–89, 2007.

KROSNICK, J. A. Survey research. *Annual review of psychology*, Annual Reviews 4139 El Camino Way, PO Box 10139, Palo Alto, CA 94303-0139, USA, v. 50, n. 1, p. 537–567, 1999.

LETHBRIDGE, T. C. A survey of the relevance of computer science and software engineering education. In: IEEE. *Proceedings 11th Conference on Software Engineering Education*. [S.l.], 1998. p. 56–66.

MAGHER, M. How to make a conceptual framework for a thesis. *Retrieved November*, 2016.

MALIK, B.; ZAFAR, S. et al. A systematic mapping study on software engineering education. In: WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY (WASET). *Proceedings of World Academy of Science, Engineering and Technology*. [S.l.], 2012. v. 6, n. 11, p. 3343–3353.

MARCH, S. T.; SMITH, G. F. Design and natural science research on information technology. *Decision support systems*, Elsevier, v. 15, n. 4, p. 251–266, 1995.

MARQUES, M. R.; QUISPE, A.; OCHOA, S. F. A systematic mapping study on practical approaches to teaching software engineering. In: IEEE. *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. [S.l.], 2014. p. 1–8.

Marques, M. R.; Quispe, A.; Ochoa, S. F. A systematic mapping study on practical approaches to teaching software engineering. In: *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. [S.l.: s.n.], 2014. p. 1–8.

MAYO, W. P.; DONNELLY, M. B.; SCHWARTZ, R. W. Characteristics of the ideal problem-based learning tutor in clinical medicine. *Evaluation & the Health Professions*, SAGE PUBLICATIONS, INC. 2455 Teller Road, Thousand Oaks, CA 91320, v. 18, n. 2, p. 124–136, 1995.

MEIRA, S. Sistemas de informação e engenharia de software–cadê as escolas. *Revista da SBC Engenharia de Software-Qual é o impacto da ES no mercado de Computação e na sociedade como um todo*, p. 11–15, 2015.

MERGEN, S.; KEPLER, F.; SILVA, J. P. S. da; CERA, M. C. Using pdca as a general framework for teaching and evaluating the learning of software engineering disciplines. In: *Anais do IX Simpósio Brasileiro de Sistemas de Informação*. Porto Alegre, RS, Brasil: SBC, 2013. p. 451–462. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbsi/article/view/5711>.

Meyer, B. Software engineering in the academy. *Computer*, v. 34, n. 5, p. 28–35, May 2001. ISSN 0018-9162.

MIZUKAMI, M. d. G. N. *Ensino: as abordagens do processo*. [S.l.]: Editora Pedagógica e Universitária São Paulo, 1986.

MOEN, R.; NORMAN, C. *Evolution of the PDCA cycle*. [S.l.]: Citeseer, 2006.

NASCIMENTO, D. M.; COX, K.; ALMEIDA, T.; SAMPAIO, W.; BITTENCOURT, R. A.; SOUZA, R.; CHAVEZ, C. Using open source projects in software engineering education: A systematic mapping study. In: IEEE. *2013 IEEE Frontiers in Education Conference (FIE)*. [S.l.], 2013. p. 1837–1843.

Nauman, M.; Uzair, M. In: *20th Conference on Software Engineering Education Training (CSEET'07)*. [S.l.: s.n.], 2007.

NICHOLS, M.; CATOR, K. *Challenge Based Learning. White Paper. Cupertino, California: Apple*. [S.l.]: Inc, 2008.

NIKNEJAD, N.; AMIRI, I. S. et al. Introduction of service-oriented architecture (soa) adoption. In: *The Impact of Service Oriented Architecture Adoption on Organizations*. [S.l.]: Springer, 2019. p. 1–8.

NUSEIBEH, B.; EASTERBROOK, S. Requirements engineering: a roadmap. In: ACM. *Proceedings of the Conference on the Future of Software Engineering*. [S.l.], 2000. p. 35–46.

OCHODEK, M. A scrum-centric framework for organizing software engineering academic courses. In: *Towards a Synergistic Combination of Research and Practice in Software Engineering*. [S.l.]: Springer, 2018. p. 207–220.

OLIVEIRA, A.; RODRIGUES, R.; GARCIA, V. Um mapeamento sistemático para problem based learning aplicado à ciência da computação. In: . [S.l.: s.n.], 2012.

PEIXOTO, D. C.; POSSA, R. M.; RESENDE, R. F.; PÁDUA, C. I. P. Faseng: A framework for development of software engineering simulation games. In: IEEE. *2012 Frontiers in Education Conference Proceedings*. [S.l.], 2012. p. 1–6.

PIAGET, J.; BUEY, F. F. et al. *Psicología y pedagogía*. [S.l.]: Ariel Barcelona, 1972.

PINTO, G. H. L.; FILHO, F. F.; STEINMACHER, I.; GEROSA, M. A. Training software engineers using open-source software: the professors' perspective. In: IEEE. *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. [S.l.], 2017. p. 117–121.

PORTELA, C.; VASCONCELOS, A.; OLIVEIRA, S. R. B. Frames: Uma proposta de framework para o ensino de tópicos da engenharia de software. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2016. v. 27, n. 1, p. 1361.

PRIKLADNICKI, R.; ALBUQUERQUE, A. B.; WANGENHEIM, C. G. von; CABRAL, R. Ensino de engenharia de software: desafios, estratégias de ensino e lições aprendidas. *FEES-Fórum de Educação em Engenharia de Software*, p. 1–8, 2009.

RAMOS, D. O que é e como usar o diagrama de tartaruga. *Blog da Qualidade*, 2017.

Rodrigues, A. N.; dos Santos, S. C. A framework for applying problem-based learning to computing education. In: *2016 IEEE Frontiers in Education Conference (FIE)*. [S.l.: s.n.], 2016. p. 1–7.

SAFARI, F.; SAFARI, N.; HASANZADEH, A. The adoption of software-as-a-service (saas): ranking the determinants. *Journal of Enterprise Information Management*, Emerald Group Publishing Limited, v. 28, n. 3, p. 400–422, 2015.

SANTORO, F. M.; BORGES, M. R. d. S.; SANTOS, N. Um framework para estudo de ambientes de suporte à aprendizagem cooperativa. *Revista Brasileira de Informática na Educação*, v. 4, n. 1, p. 51–68, 1999.

SANTOS, S.; ALEXANDRE, G.; RODRIGUES, A. Applying pbl in project management education: A case study of an undergraduate course. In: IEEE. *2015 IEEE Frontiers in Education Conference (FIE)*. [S.l.], 2015. p. 1–8.

SANTOS, S. C. D. O processo de ensino-aprendizagem e a relação professor-aluno: aplicação dos"sete princípios para a boa prática na educação de ensino superior". *REGE Revista de Gestão*, v. 8, n. 1, 2010.

SANTOS, S. C. D.; BATISTA, M. d. C. M.; CAVALCANTI, A. P. C.; ALBUQUERQUE, J. O.; MEIRA, S. R. Applying pbl in software engineering education. In: IEEE. *2009 22nd Conference on Software Engineering Education and Training*. [S.l.], 2009. p. 182–189.

SAVERY, J. R. Overview of problem-based learning: Definitions and distinctions. *Essential readings in problem-based learning: Exploring and extending the legacy of Howard S. Barrows*, v. 9, p. 5–15, 2015.

SAVERY, J. R.; DUFFY, T. M. Problem based learning: An instructional model and its constructivist framework. *Educational technology*, v. 35, n. 5, p. 31–38, 1995.

SCHNEIDER, J.-G.; JOHNSTON, L. extreme programming at universities-an educational perspective. In: IEEE. *25th International Conference on Software Engineering, 2003. Proceedings.* [S.l.], 2003. p. 594–599.

SILVA, L. F. da; LEITE, J. C. S. do P.; KOOGAN, K. Ensino de engenharia de software: Relato de experiências. 2004.

SOCIETY, C. C. *INTERNATIONAL ORGANIZATION FOR STANDARDIZATION/ INTERNATIONAL ELECTROTECHNICAL (ISO/IEC/IEEE) 24.765. Systems and software engineering – vocabulary*. [S.l.: s.n.], 2010.

SOMMERVILLE, I. Software engineering 9th edition. *ISBN-10137035152*, 2011.

SØRENSEN, L.; FALCH, M.; SKOUBY, K. E. Report on problem based learning for software engineering. 2018.

STEVENS, K. T. Experiences teaching software engineering for the first time. In: ACM. *ACM SIGCSE Bulletin*. [S.l.], 2001. v. 33, n. 3, p. 77–80.

TAKEDA, H.; VEERKAMP, P.; YOSHIKAWA, H. Modeling design process. *AI magazine*, v. 11, n. 4, p. 37–37, 1990.

THOMAS, J. W. A review of research on project-based learning. 2000.

TOMAYKO, J. E. *Teaching a project-intensive introduction to software engineering*. [S.l.], 1987.

TOPI, H.; VALACICH, J. S.; WRIGHT, R. T.; KAISER, K.; JR, J. F. N.; SIPIOR, J. C.; VREEDE, G.-J. de. Is 2010: Curriculum guidelines for undergraduate degree programs in information systems. *Communications of the Association for Information Systems*, v. 26, n. 1, p. 18, 2010.

VITORINO, S. M. A. Brasil. utilização do ciclo pdca e melhoria continua como ferramentas de gerenciamento escolar no distrito federal. *Aularia: Revista Digital de Comunicación*, Grupo Comunicar, v. 8, n. 1, p. 31–34, 2019.

VOSGERAU, D. S. R.; ROMANOWSKI, J. P. Review studies: conceptual and methodological implications. *Revista Diálogo Educacional*, v. 14, n. 41, p. 165–189, 2014.

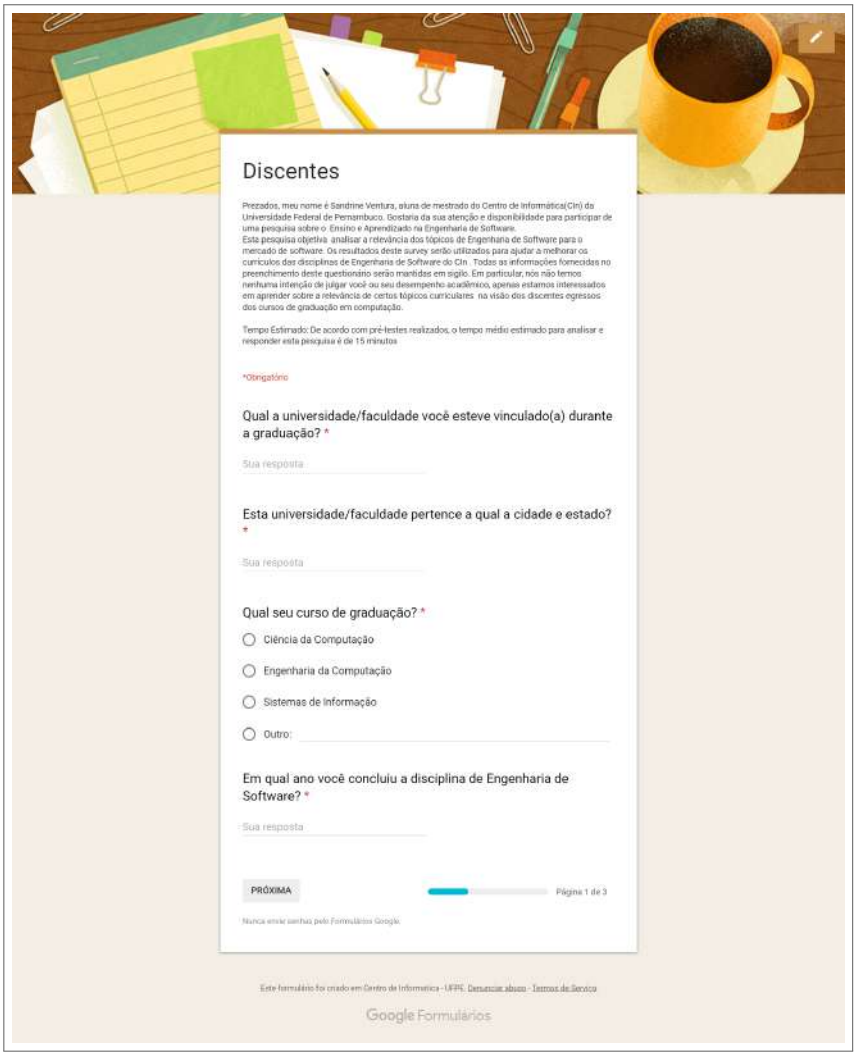
WANGENHEIM, C. G. V.; SHULL, F. To game or not to game? *IEEE software*, IEEE, v. 26, n. 2, p. 92–94, 2009.

WIBISONO, A.; NISAFANI, A. S. Curriculum structure of the undergraduate programs of information systems in indonesia in the year of 2013. *ISICO 2013*, v. 2013, 2013.

ZORZO, A.; NUNES, D.; MATOS, E.; STEINMACHER, I.; LEITE, J.; ARAUJO, R.; CORREIA, R.; MARTINS, S. Referenciais de formação para os cursos de graduação em computação. sociedade brasileira de computação (sbc). 153p, 2017. 2017.

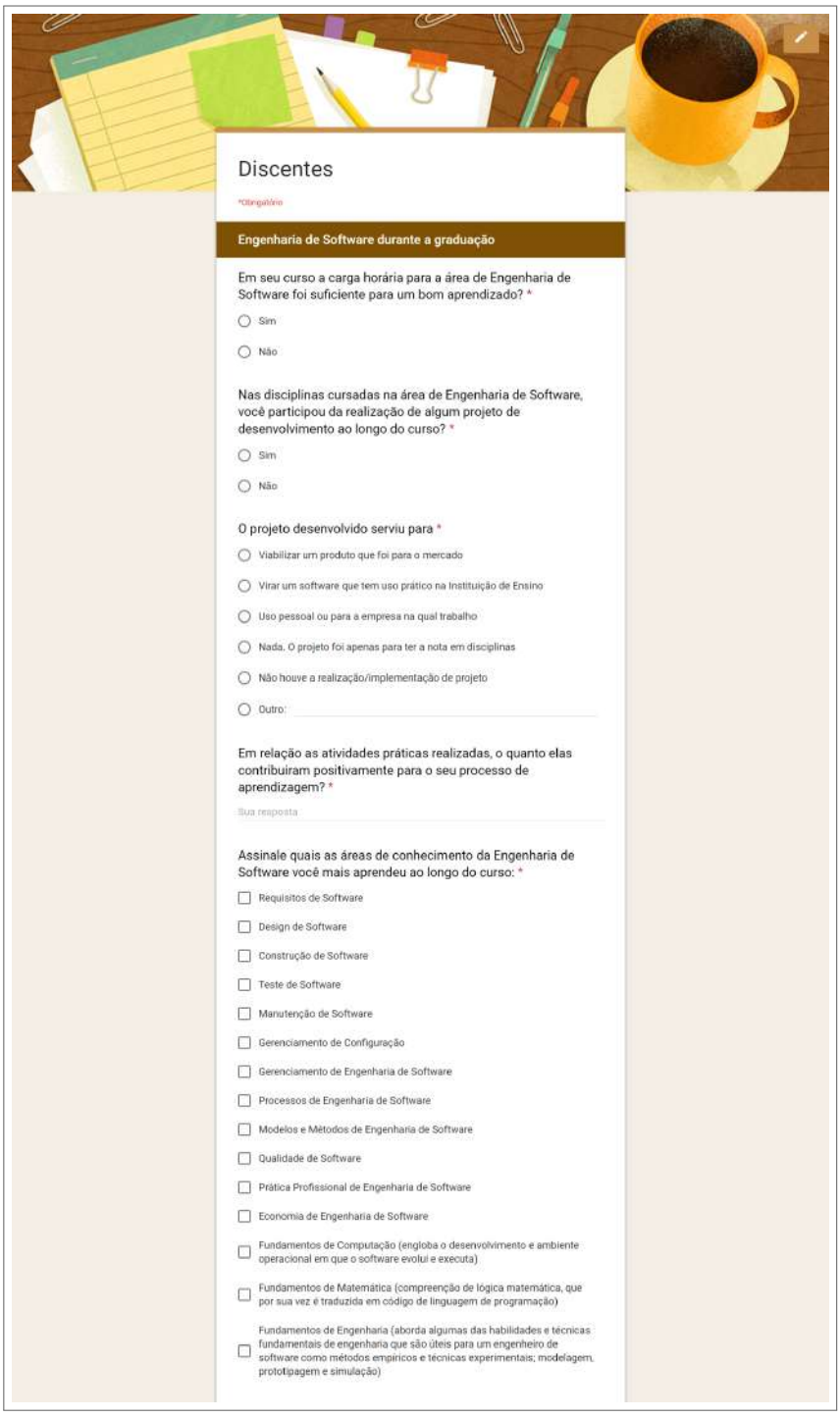
APÊNDICE A – *TEMPLATE* QUESTÕES DO SURVEY PARA OS DISCENTES

Figura 33 – Survey discentes



Fonte: Elaborado pela autora (2019).

Figura 34 – Continuação - Survey discentes



Discentes

*Obrigatório

Engenharia de Software durante a graduação

Em seu curso a carga horária para a área de Engenharia de Software foi suficiente para um bom aprendizado? *

- Sim
- Não

Nas disciplinas cursadas na área de Engenharia de Software, você participou da realização de algum projeto de desenvolvimento ao longo do curso? *

- Sim
- Não

O projeto desenvolvido serviu para *

- Viabilizar um produto que foi para o mercado
- Virar um software que tem uso prático na Instituição de Ensino
- Uso pessoal ou para a empresa na qual trabalho
- Nada. O projeto foi apenas para ter a nota em disciplinas
- Não houve a realização/implementação de projeto
- Outro:

Em relação as atividades práticas realizadas, o quanto elas contribuiram positivamente para o seu processo de aprendizagem? *

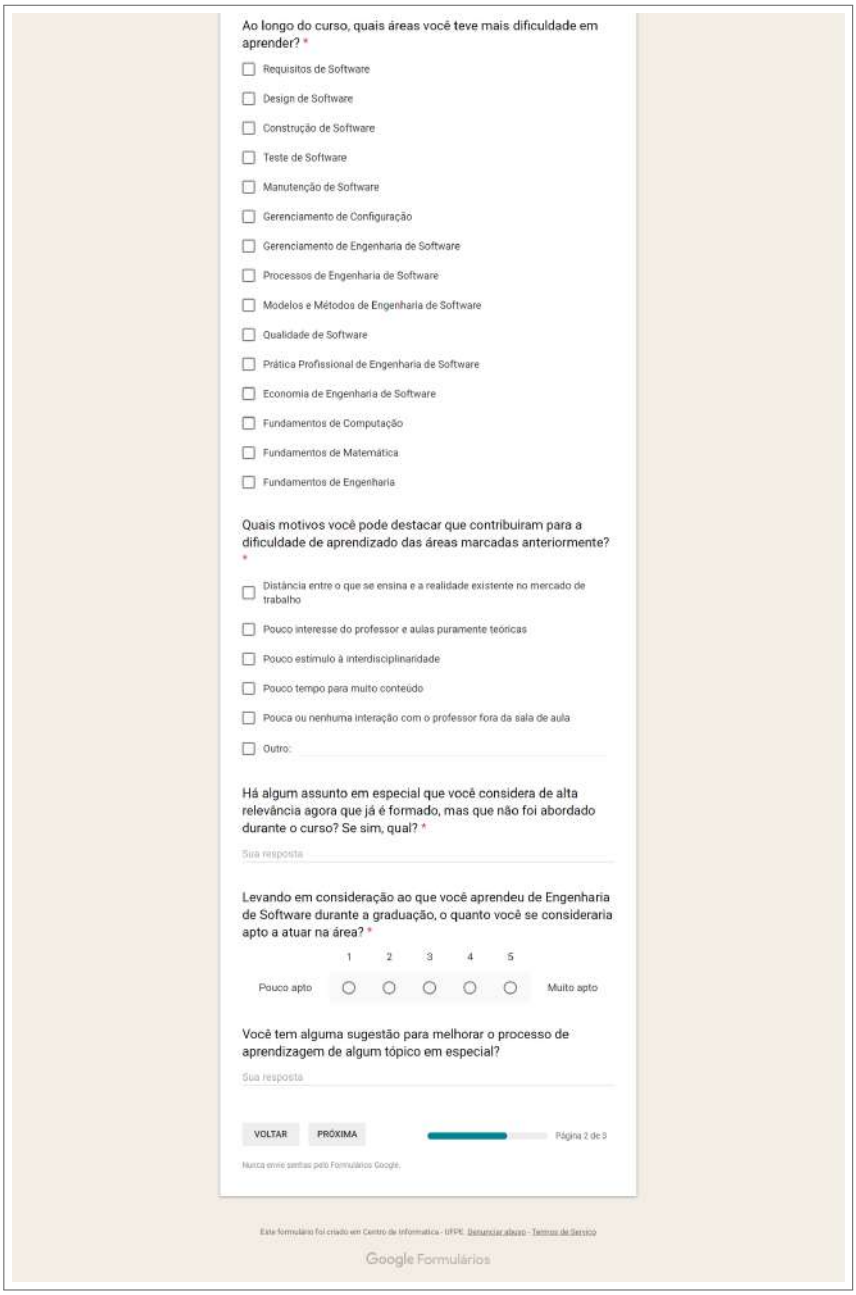
Sua resposta

Assinale quais as áreas de conhecimento da Engenharia de Software você mais aprendeu ao longo do curso: *

- [ ] Requisitos de Software
- [ ] Design de Software
- [ ] Construção de Software
- [ ] Teste de Software
- [ ] Manutenção de Software
- [ ] Gerenciamento de Configuração
- [ ] Gerenciamento de Engenharia de Software
- [ ] Processos de Engenharia de Software
- [ ] Modelos e Métodos de Engenharia de Software
- [ ] Qualidade de Software
- [ ] Prática Profissional de Engenharia de Software
- [ ] Economia de Engenharia de Software
- [ ] Fundamentos de Computação (engloba o desenvolvimento e ambiente operacional em que o software evolui e executa)
- [ ] Fundamentos de Matemática (compreenção de lógica matemática, que por sua vez é traduzida em código de linguagem de programação)
- [ ] Fundamentos de Engenharia (aborda algumas das habilidades e técnicas fundamentais de engenharia que são úteis para um engenheiro de software como métodos empíricos e técnicas experimentais; modelagem, prototipagem e simulação)

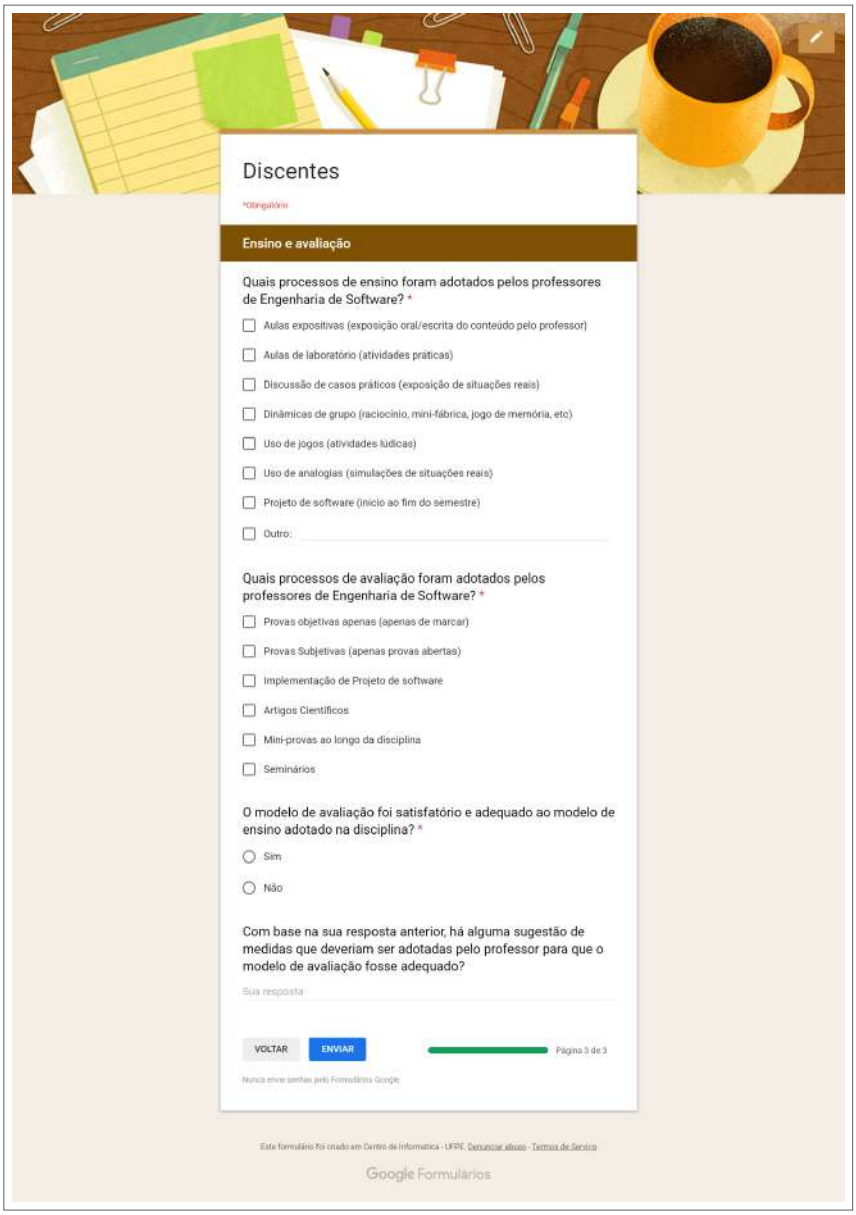
Fonte: Elaborado pela autora (2019).

Figura 35 – Continuação - Survey discentes



Fonte: Elaborado pela autora (2019).

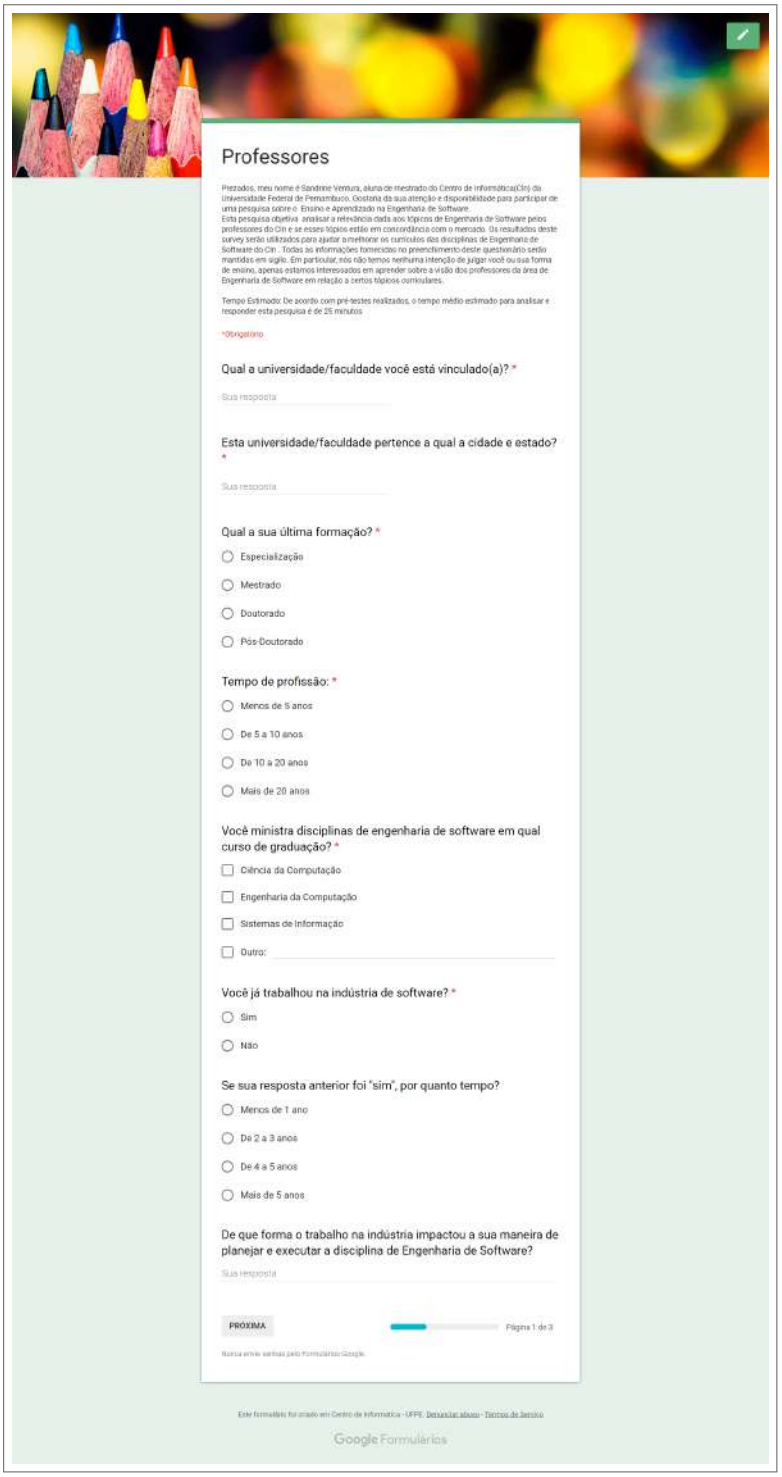
Figura 36 – Continuação - Survey discentes



Fonte: Elaborado pela autora (2019).

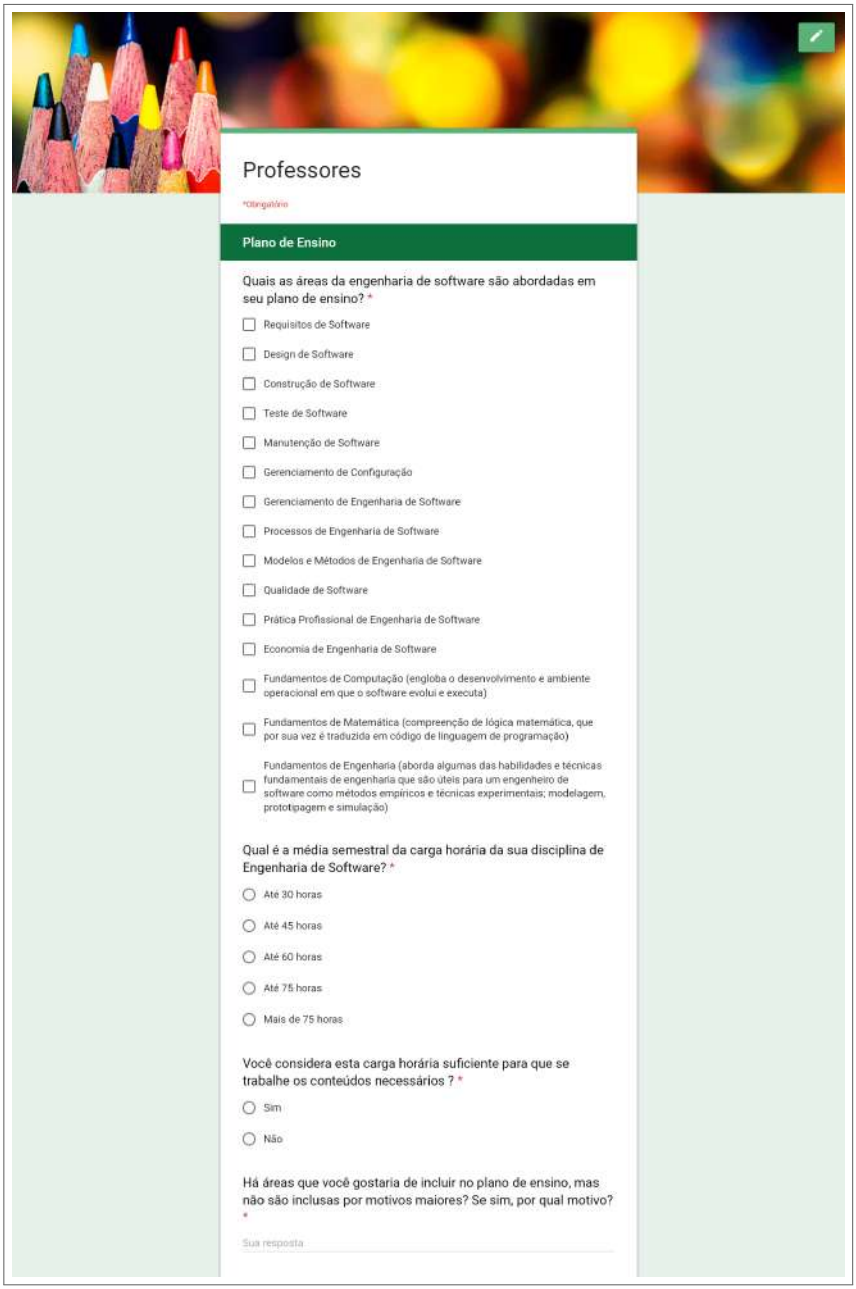
APÊNDICE B – *TEMPLATE* QUESTÕES DO SURVEY PARA OS PROFESSORES

Figura 37 – Survey professores



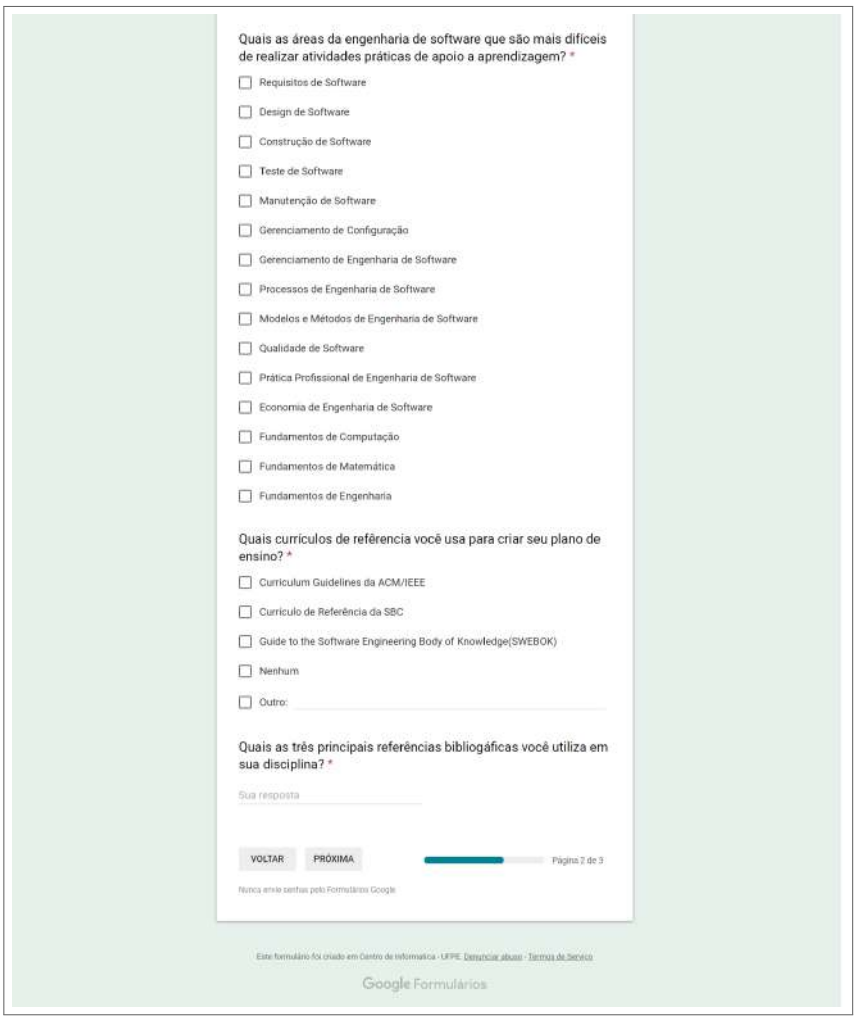
Fonte: Elaborado pela autora (2019).

Figura 38 – Continuação - Survey professores



Fonte: Elaborado pela autora (2019).

Figura 39 – Continuação - Survey professores

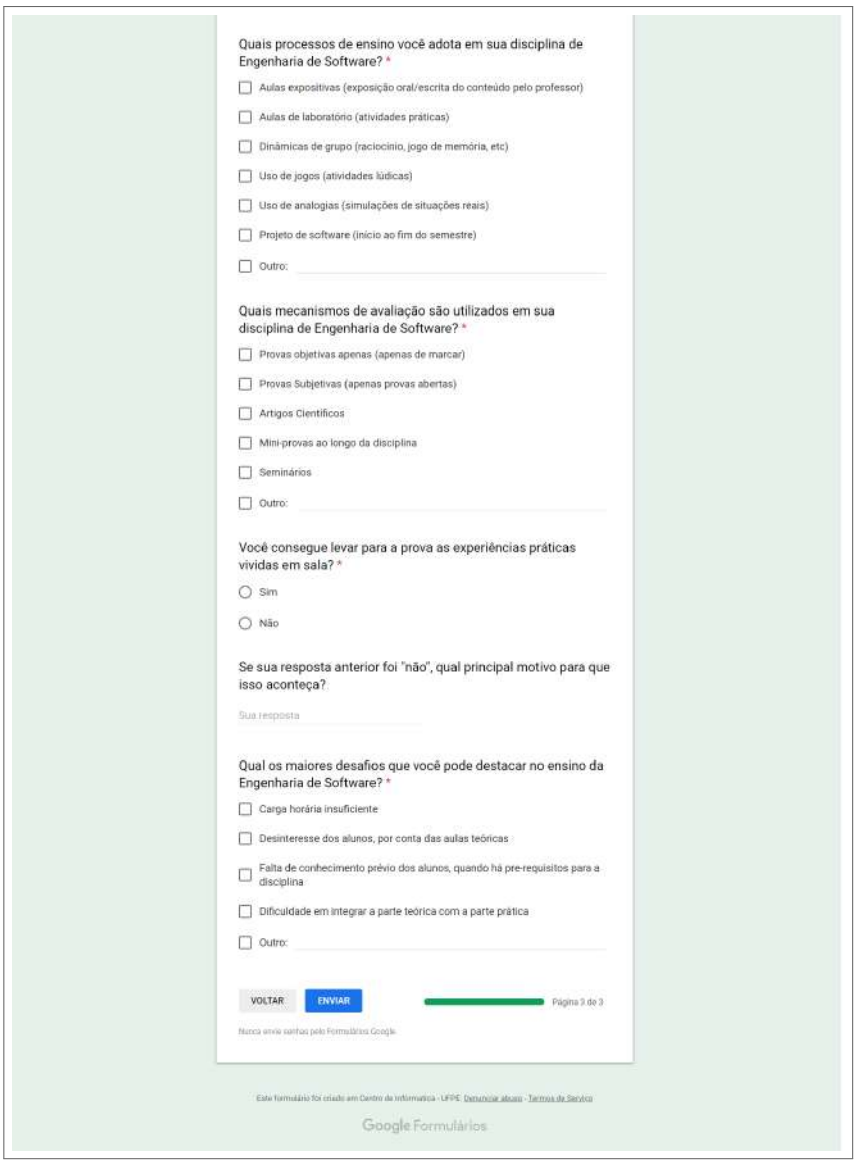


Fonte: Elaborado pela autora (2019).

Figura 40 – Continuação - Survey professores

Fonte: Elaborado pela autora (2019).

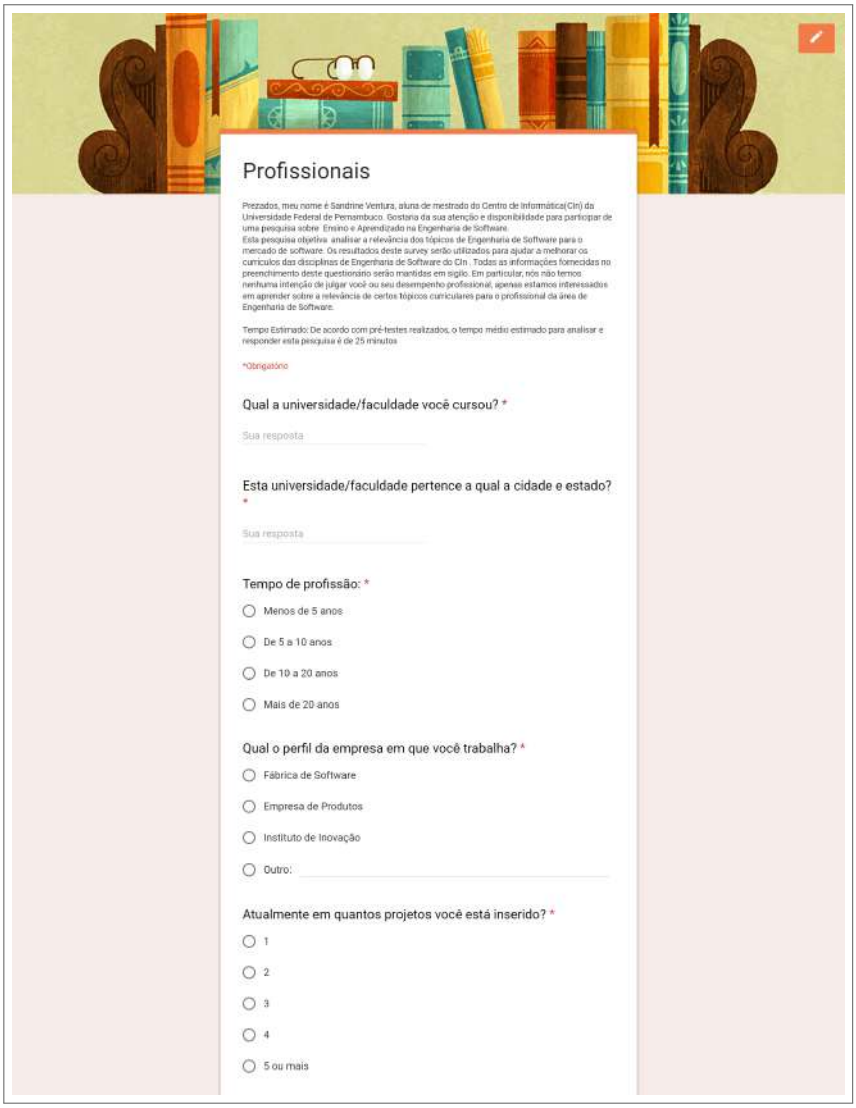
Figura 41 – Continuação - Survey professores



Fonte: Elaborado pela autora (2019).

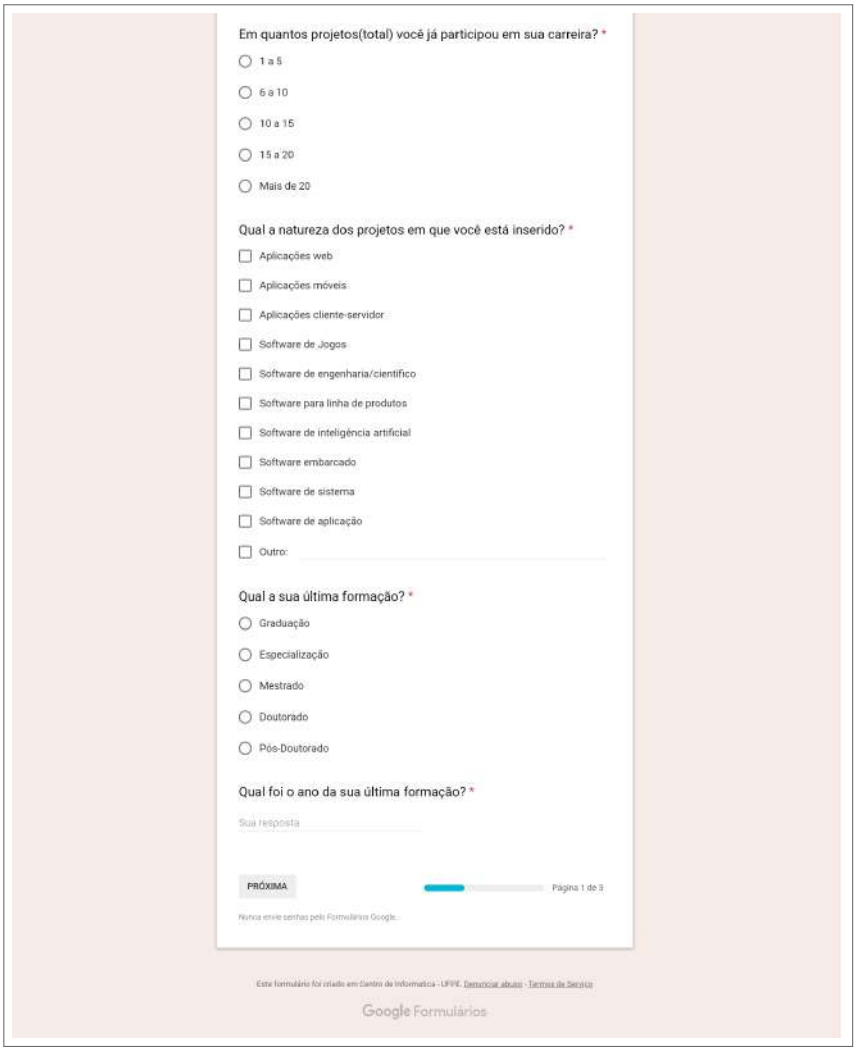
APÊNDICE C – *TEMPLATE* QUESTÕES DO SURVEY PARA OS PROFISSIONAIS

Figura 42 – Survey profissionais



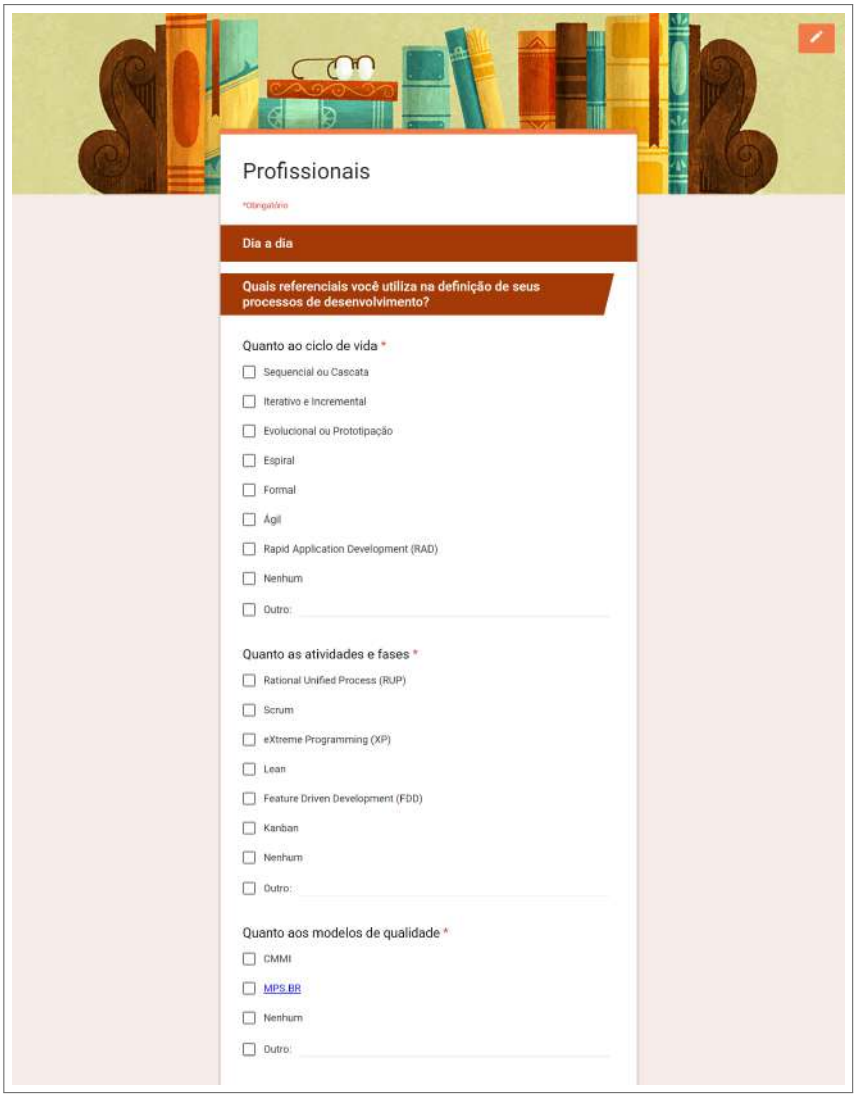
Fonte: Elaborado pela autora (2019).

Figura 43 – Continuação - Survey profissionais



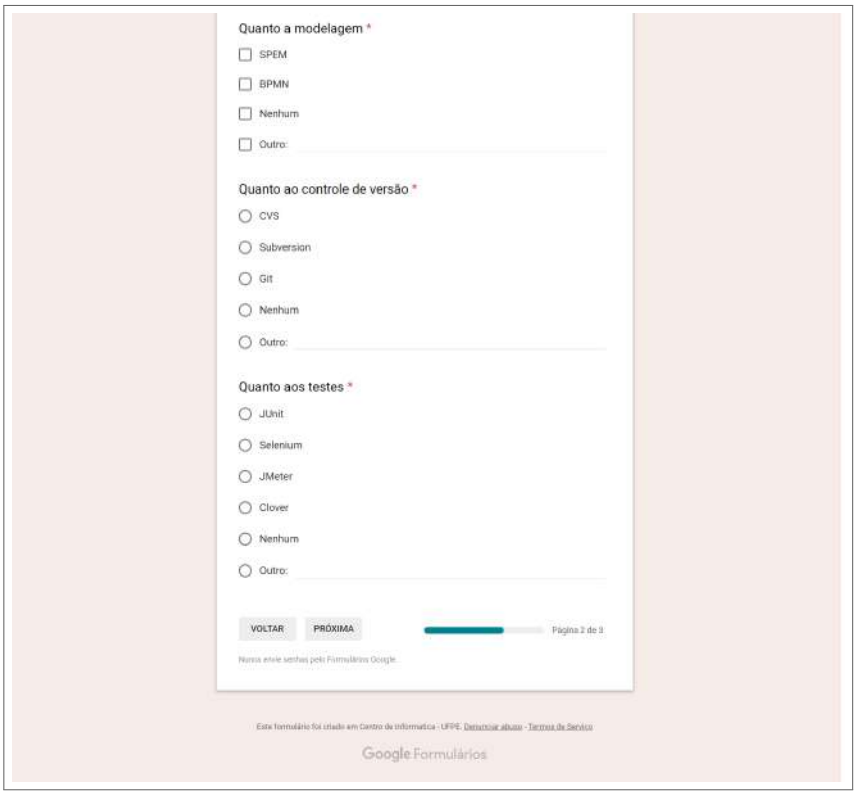
Fonte: Elaborado pela autora (2019).

Figura 44 – Continuação - Survey profissionais



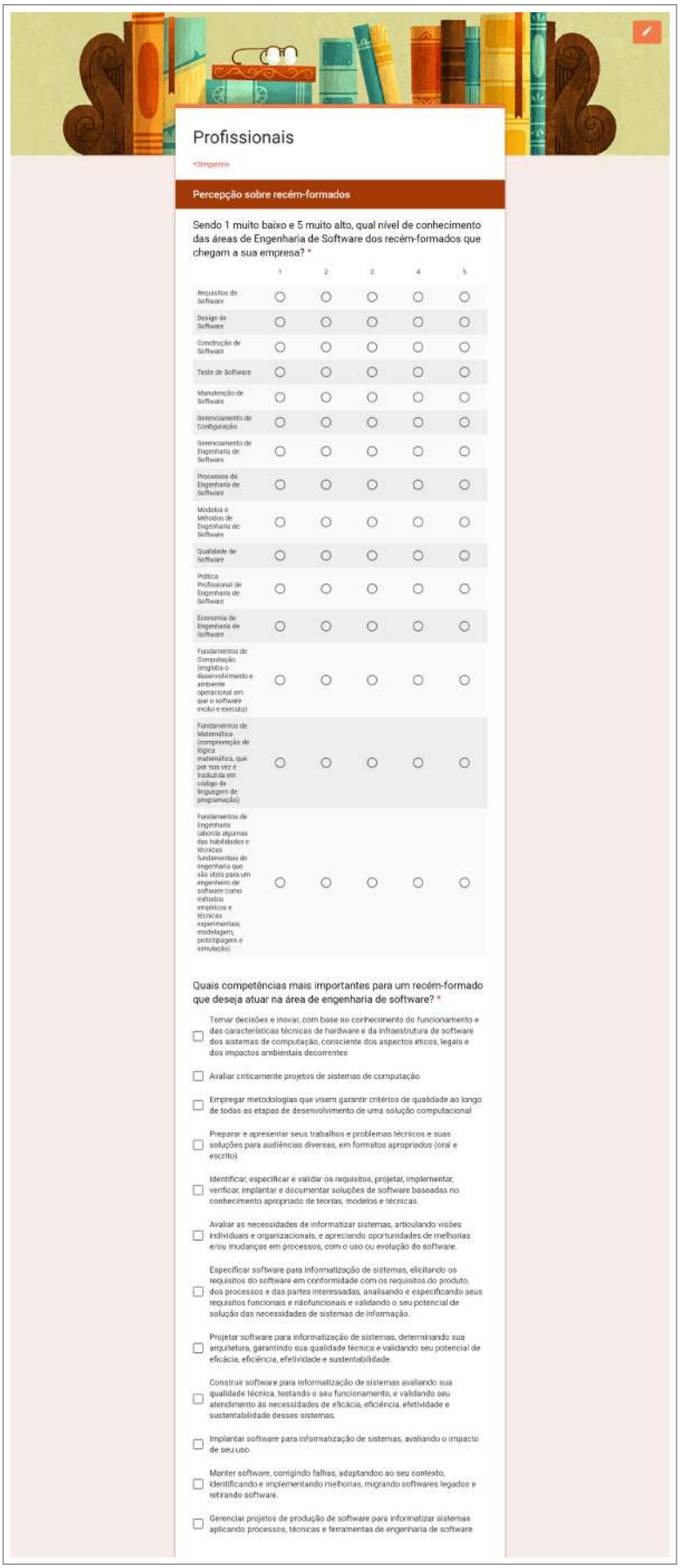
Fonte: Elaborado pela autora (2019).

Figura 45 – Continuação - Survey profissionais



Fonte: Elaborado pela autora (2019).

Figura 46 – Continuação - Survey profissionais



Profissionais

*Obrigatório

Percepção sobre recém-formados

Sendo 1 muito baixo e 5 muito alto, qual nível de conhecimento das áreas de Engenharia de Software dos recém-formados que chegam a sua empresa? *

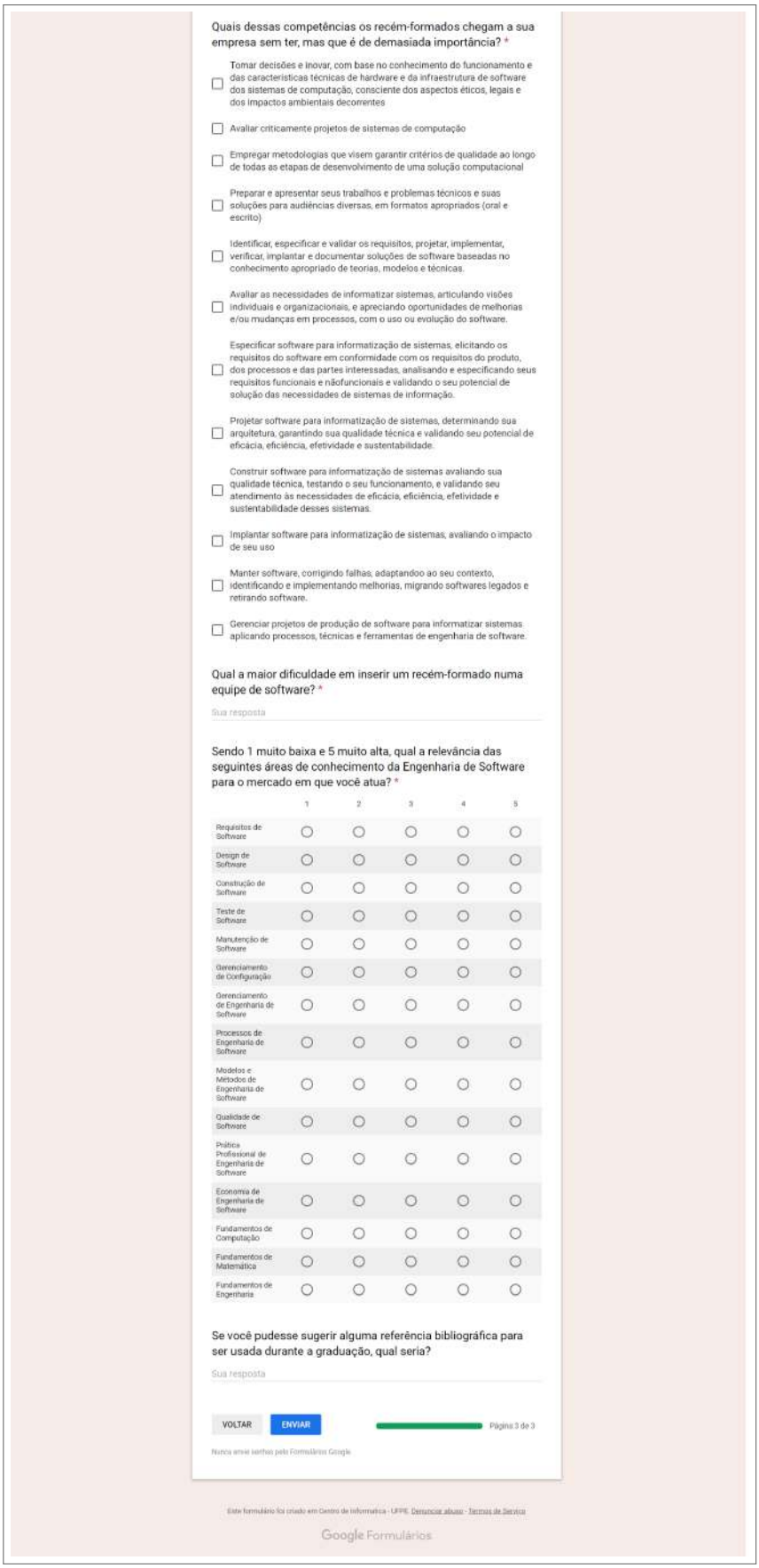
	1	2	3	4	5
Requisitos de Software	○	○	○	○	○
Design de Software	○	○	○	○	○
Construção de Software	○	○	○	○	○
Teste de Software	○	○	○	○	○
Manutenção de Software	○	○	○	○	○
Gerenciamento de Configuração	○	○	○	○	○
Gerenciamento de Engenharia de Software	○	○	○	○	○
Processos de Engenharia de Software	○	○	○	○	○
Modelos e Métodos de Engenharia de Software	○	○	○	○	○
Qualidade de Software	○	○	○	○	○
Prática Profissional de Engenharia de Software	○	○	○	○	○
Economia de Engenharia de Software	○	○	○	○	○
Fundamentos de Computação (engloba o desenvolvimento e ambiente operacional em que o software é criado e executa)	○	○	○	○	○
Fundamentos de Matemática (compreensão de lógica matemática, que por sua vez é traduzida em código de linguagem de programação)	○	○	○	○	○
Fundamentos de Engenharia (abordar algumas das habilidades e técnicas fundamentais de engenharia que são úteis para um engenheiro de software como: métodos empíricos e técnicas experimentais; modelagem, prototipagem e simulação).	○	○	○	○	○

Quais competências mais importantes para um recém-formado que deseja atuar na área de engenharia de software? *

- ☐ Tomar decisões e inovar, com base no conhecimento do funcionamento e das características técnicas de hardware e da infraestrutura de software dos sistemas de computação, consciente dos aspectos éticos, legais e dos impactos ambientais decorrentes
- ☐ Avaliar criticamente projetos de sistemas de computação
- ☐ Empregar metodologias que visem garantir critérios de qualidade ao longo de todas as etapas de desenvolvimento de uma solução computacional
- ☐ Preparar e apresentar seus trabalhos e problemas técnicos e suas soluções para audiências diversas, em formatos apropriados (oral e escrito)
- ☐ Identificar, especificar e validar os requisitos, projetar, implementar, verificar, implantar e documentar soluções de software baseadas no conhecimento apropriado de teorias, modelos e técnicas.
- ☐ Avaliar as necessidades de informatizar sistemas, articulando visões individuais e organizacionais, e apreciando oportunidades de melhorias e/ou mudanças em processos, com o uso ou evolução do software.
- ☐ Especificar software para informatização de sistemas, elicitando os requisitos do software em conformidade com os requisitos do produto, dos processos e das partes interessadas, analisando e especificando seus requisitos funcionais e não-funcionais e validando o seu potencial de solução das necessidades de sistemas de informação.
- ☐ Projetar software para informatização de sistemas, determinando sua arquitetura, garantindo sua qualidade técnica e validando seu potencial de eficácia, eficiência, efetividade e sustentabilidade.
- ☐ Construir software para informatização de sistemas avaliando sua qualidade técnica, testando o seu funcionamento, e validando seu atendimento às necessidades de eficácia, eficiência, efetividade e sustentabilidade desses sistemas.
- ☐ Implantar software para informatização de sistemas, avaliando o impacto de seu uso.
- ☐ Manter software, corrigindo falhas, adaptando-o ao seu contexto, identificando e implementando melhorias, migrando softwares legados e retirando software.
- ☐ Gerenciar projetos de produção de software para informatizar sistemas aplicando processos, técnicas e ferramentas de engenharia de software.

Fonte: Elaborado pela autora (2019).

Figura 47 – Continuação - Survey profissionais



Quais dessas competências os recém-formados chegam a sua empresa sem ter, mas que é de demasiada importância? *

- [ ] Tomar decisões e inovar, com base no conhecimento do funcionamento e das características técnicas de hardware e da infraestrutura de software dos sistemas de computação, consciente dos aspectos éticos, legais e dos impactos ambientais decorrentes
- [ ] Avaliar criticamente projetos de sistemas de computação
- [ ] Empregar metodologias que visem garantir critérios de qualidade ao longo de todas as etapas de desenvolvimento de uma solução computacional
- [ ] Preparar e apresentar seus trabalhos e problemas técnicos e suas soluções para audiências diversas, em formatos apropriados (oral e escrito)
- [ ] Identificar, especificar e validar os requisitos, projetar, implementar, verificar, implantar e documentar soluções de software baseadas no conhecimento apropriado de teorias, modelos e técnicas.
- [ ] Avaliar as necessidades de informatizar sistemas, articulando visões individuais e organizacionais, e apreciando oportunidades de melhorias e/ou mudanças em processos, com o uso ou evolução do software.
- [ ] Especificar software para informatização de sistemas, elicitando os requisitos do software em conformidade com os requisitos do produto, dos processos e das partes interessadas, analisando e especificando seus requisitos funcionais e nãofuncionais e validando o seu potencial de solução das necessidades de sistemas de informação.
- [ ] Projetar software para informatização de sistemas, determinando sua arquitetura, garantindo sua qualidade técnica e validando seu potencial de eficácia, eficiência, efetividade e sustentabilidade.
- [ ] Construir software para informatização de sistemas avaliando sua qualidade técnica, testando o seu funcionamento, e validando seu atendimento às necessidades de eficácia, eficiência, efetividade e sustentabilidade desses sistemas.
- [ ] Implantar software para informatização de sistemas, avaliando o impacto de seu uso
- [ ] Manter software, corrigindo falhas, adaptandoo ao seu contexto, identificando e implementando melhorias, migrando softwares legados e retirando software.
- [ ] Gerenciar projetos de produção de software para informatizar sistemas aplicando processos, técnicas e ferramentas de engenharia de software.

Qual a maior dificuldade em inserir um recém-formado numa equipe de software? *

Sua resposta

Sendo 1 muito baixa e 5 muito alta, qual a relevância das seguintes áreas de conhecimento da Engenharia de Software para o mercado em que você atua? *

	1	2	3	4	5
Requisitos de Software	○	○	○	○	○
Design de Software	○	○	○	○	○
Construção de Software	○	○	○	○	○
Teste de Software	○	○	○	○	○
Manutenção de Software	○	○	○	○	○
Gerenciamento de Configuração	○	○	○	○	○
Gerenciamento de Engenharia de Software	○	○	○	○	○
Processos de Engenharia de Software	○	○	○	○	○
Modelos e Métodos de Engenharia de Software	○	○	○	○	○
Qualidade de Software	○	○	○	○	○
Prática Profissional de Engenharia de Software	○	○	○	○	○
Economia de Engenharia de Software	○	○	○	○	○
Fundamentos de Computação	○	○	○	○	○
Fundamentos de Matemática	○	○	○	○	○
Fundamentos de Engenharia	○	○	○	○	○

Se você pudesse sugerir alguma referência bibliográfica para ser usada durante a graduação, qual seria?

Sua resposta

VOLTAR ENVIAR Página 3 de 3

Nunca envie senhas pelo Formulários Google.

Este formulário foi criado em Centro de Informática - UFPE. Denunciar abuso - Termos de Serviço

Google Formulários

Fonte: Elaborado pela autora (2019).

APÊNDICE D – ANÁLISE DE VALIDADE DE CONSTRUÇÃO

Para analisar esta questão, foi feita uma comparação e classificação dos tópicos ordenados pela importância da pesquisa aqui mostrada, com a classificação das pesquisas realizadas por (LETHBRIDGE, 1998) (KITCHENHAM et al., 2005). Entretanto, é importante ressaltar que a importância foi avaliada de forma diferente em cada uma das pesquisas. No presente estudo, a importância foi calculada sobre a proporção de respostas dos participantes, levando em consideração as áreas de conhecimento que receberam mais de 50% de votos.

No estudo de (KITCHENHAM et al., 2005), a importância foi calculada sobre a proporção de respostas dos participantes, que receberam três pontos (indicando, que a utilidade do tópico era no mínimo moderada) ou mais para a questão 3.

No estudo de (LETHBRIDGE, 1998), várias maneiras de avaliar a importância foram aplicadas. No entanto, a apresentação da lista completa dos tópicos está disposta pelo cálculo do valor médio das respostas das questões 3 e 4.

Outro aspecto a ser considerado é a variação dos tópicos considerados em cada uma das pesquisas. Em razão da quantidade diferente de tópicos nas pesquisas, foram considerados os tópicos de engenharia de software que estavam entre os 5 mais importantes da pesquisa de (LETHBRIDGE, 1998), e os tópicos que estavam entre os 5 mais importantes com relação aos resultados aqui apresentados, e os de (KITCHENHAM et al., 2005). Estes tópicos são apresentados na Tabela 23.

Tabela 23 – Importância dos tópicos em relação a outras pesquisas

(**LETHBRIDGE, 1998**)	(**KITCHENHAM et al., 2005**)	
Design de software geral	Concepção essencial de subsistema	Requisitos de software
Métodos de engenharia de software	Gerenciamento de software	Processos de ES
Gerenciamento de software	Métodos de engenharia de software	Teste de software
Concepção essencial de subsistema	Design de software geral	Construção de software
Técnicas de aplicação especializadas	Técnicas de aplicação especializadas	Design de software