Pós-Graduação em Ciência da Computação

CARLOS DOS SANTOS PORTELA

UM MODELO ITERATIVO PARA O ENSINO DE ENGENHARIA DE SOFTWARE BASEADO EM ABORDAGENS FOCADAS NO ALUNO E PRÁTICAS DE CAPACITAÇÃO DA INDÚSTRIA

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

Recife
2017

Carlos dos Santos Portela

Um Modelo Iterativo para o Ensino de Engenharia de Software Baseado em Abordagens Focadas no Aluno e Práticas de Capacitação da Indústria

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Área de concentração: Engenharia de Software.

Orientador: Profº. Dr. Alexandre Marcos Lins De Vasconcelos
Coorientador: Profº. Dr. Sandro Ronaldo Bezerra Oliveira

Recife
2017

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

P843m Portela, Carlos dos Santos
Um modelo iterativo para o ensino de engenharia de software baseado em abordagens focadas no aluno e práticas de capacitação da indústria / Carlos dos Santos Portela. – 2017.
285 f.: il., fig., tab.

Orientador: Alexandre Marcos Lins de Vasconcelos.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências e apêndices.

1. Engenharia de software. 2. Modelo de ensino. I. Vasconcelos, Alexandre Marcos Lins de (orientador). II. Título.

005.1 CDD (23. ed.) UFPE- MEI 2018-083

Carlos dos Santos Portela

Um Modelo Iterativo para o Ensino de Engenharia de Software Baseado em Abordagens Focadas no Aluno e Práticas de Capacitação da Indústria

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

Aprovado em: 30/10/2017.

_____

Orientador: Prof. Dr. Alexandre Marcos Lins de Vasconcelos

BANCA EXAMINADORA

_____

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

_____

Prof. Dr. Vinicius Cardoso Garcia
Centro de Informática / UFPE

_____

Profa. Dra. Ingrid Oliveira de Nunes
Instituto de Informática/UFRGS

_____

Prof. Dr. Adriano Bessa Albuquerque
Mestrado em Informática Aplicada/UNIFOR

_____

Profa. Dra. Renata Teles Moreira
Departamento de Ciência da Computação/ UFLA

Dedico esta tese à minha esposa Juliana Pennafort
e ao nosso filho Mateus Pennafort Portela.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por me guiar nesta jornada de 4 anos de aprendizagens e descobertas. Obrigado por me ensinar a ser paciente nos momentos difíceis e por colocar as pessoas certas (professores e amigos doutorandos) no meu caminho, as quais me ajudaram em cada etapa deste doutorado. Agradeço imensamente todas as bênçãos alcançadas durante esse período do doutorado.

Aos meus pais, Luzia e Raimundo Portela, que desde cedo me ensinaram que o estudo era o melhor caminho para mudança de vida. Em especial, agradeço a minha esposa, Juliana Pennafort, que conheci no período de seleção do doutorado e, desde então, foi a principal incentivadora da realização deste projeto de pesquisa. Obrigado pelo melhor presente do mundo, nosso filho Mateus.

A Sandro Oliveira, pela sua amizade e por sempre acreditar em minhas decisões. Obrigado por me orientar na graduação (2009), mestrado (2012) e doutorado (2017). A Alexandre Vasconcelos, pelo seu profissionalismo e pela orientação em cada etapa dessa pesquisa. Obrigado pela confiança e por todo apoio durante o doutorado.

Aos grandes amigos que cursaram comigo as cadeiras do doutorado no CIn/UFPE e fizeram esse período em Recife-PE inesquecível. Andreza Leite, minha amiga paraense que me atura desde 2010. Wylliams Barbosa, pelas corridas e parcerias fora da universidade. Juliana Dantas, pelo apoio e macaxeiras com charque no fim de longas tardes de pesquisa. Esses 3 foram a melhor panelinha que eu poderia ter no doutorado. Adicionalmente, agradeço a meu amigo joia, Maurício Ronny pela convivência, conversas e contribuições com essa pesquisa. Wilton Oliveira e Júlio Damasceno, meus companheiros desde o mestrado, que dividiram o apartamento e as contas em Recife.

O papel fundamental de um professor não é entregar informação e sim guiar o processo de aprendizagem. O trabalho de um professor é inspirar, desafiar e motivar seus alunos a querer aprender (MULLER, 2014).

RESUMO

Tipicamente, os profissionais da indústria de software cursam graduação na área de Computação a fim de se prepararem para atuar na área de ES. Apesar disso, a indústria de software apresenta insatisfação quanto ao nível de preparação dos profissionais recém-formados. Essa carência na formação pode ser resultado de uma educação inadequada. Existe um consenso entre os pesquisadores da área de que abordagens práticas são as mais indicadas para o ensino de ES. No entanto, nem todos os cursos oferecem essa oportunidade aos alunos de realizarem atividades práticas. Nesse contexto, o principal objetivo dessa pesquisa é apoiar a adoção de abordagens práticas no processo de ensino-aprendizagem de ES a fim de que os alunos desenvolvam determinadas competências técnicas em nível de aplicação. Para tal, definiu-se um modelo iterativo que integra as principais abordagens focadas no aluno que são aplicadas no ensino de ES. Como diferencial, incorporou-se nesse modelo práticas de capacitação adotadas pela indústria de software adaptadas para o contexto acadêmico. A avaliação da documentação e usabilidade desse modelo foi feita através de um painel de especialistas composto por três professores doutores que atuam na área de ensino de ES. Posteriormente, realizou-se um experimento controlado em duas turmas de graduação com o intuito de comparar o uso do modelo iterativo e as abordagens tradicionais de ensino em relação ao desenvolvimento de competências em ES. Os dados foram coletados através de questionários estruturados e analisados usando ANOVA. O painel de especialistas considerou a documentação do modelo completa e consistente. Quanto ao experimento, os resultados obtidos na instanciação do modelo no ensino da unidade de conhecimento Gerenciamento de Projetos foram mais efetivos que os obtidos na aplicação da abordagem tradicional. No entanto, esses resultados não foram observados no experimento relacionado à unidade de Engenharia de Requisitos. Essa pesquisa identificou evidências de que a adoção de abordagens focadas no aluno e práticas de capacitação da indústria tendem a desenvolver determinadas competências técnicas em ES de maneira mais efetiva do que a abordagem tradicional de ensino. Contudo, observou-se que a motivação e o comprometimento dos alunos possuem influência direta sob esses resultados.

Palavras-chave: Ensino de Engenharia de Software. Desenvolvimento de Competências. Modelo de Ensino. Abordagens Focadas no Aluno. Práticas de Capacitação da Indústria.

ABSTRACT

Typically, software industry professionals graduate in Computing undergraduate courses in order to prepare themselves to work in the Software Engineering area. In spite of it, the software industry is not satisfied with the preparation level of the new graduate professionals. This deficiency in the professional formation may be the result of inadequate education. The fact that practical approaches are best suited for teaching Software Engineering is a consensus among researchers in the area. However, most courses do not offer to the students the opportunity to perform practical activities. Given this background, the main objective of this research is to support the adoption of practical approaches in the teaching-learning process of SE in order to the students to develop certain technical skills at the application level. With this goal, we have defined an iterative model that integrates the main approaches focused on the student applied in the SE teaching. As a differential, we incorporated in this model the training practices adopted by the software industry and adapted them to the academic context. A specialist's panel composed of three Ph.D. professors and researchers in the SE teaching area took care of the evaluation of the documentation and the usability of this model. Subsequently, we carried out a controlled experiment in two undergraduate courses in order to compare the use of this iterative model and the traditional teaching approaches in relation to the development of Software Engineering competencies. The data were collected from structured questionnaires and analyzed using ANOVA. The specialist's considered the model's documentation complete and consistent. As for the controlled experiment, the results obtained in the model instantiation in the teaching of the Software Project Management knowledge unit were more effective than the obtained in the traditional approaches application. However, these results were not observed in the experiment related to the Requirements Engineering knowledge unit. This research found evidence that the adoption of student-focused approaches and industry training practices allow the development of certain SE technical skills more effectively than the traditional teaching approach. However, it is observed that students' motivation and commitment have a direct influence on these results.

Keywords: Software Engineering Teaching. Competencies Development. Teaching Model. Student-Focused Approaches. Industry Training Practices.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABES	Associação Brasileira das Empresas de Software
ACM	*Association for Computing Machinery*
CMMI-DEV	*Capability Maturity Model Integration for Development*
ES	Engenharia de Software
IEEE	*Institute for Electrical and Electronic Engineers*
IES	Instituições de Ensino Superior
MR-MPS-SW	Modelo de Referência MPS para Software
PBL	*Problem Based Learning*
SBC	Sociedade Brasileira de Computação
SWECOM	*Software Engineering Competency Model*
TCLE	Termo de Comprometimento Livre e Esclarecido

SUMÁRIO

1 INTRODUÇÃO

A Engenharia de Software (ES) pode ser definida como “a aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção de software” (IEEE COMPUTER SOCIETY, 2014B). De acordo com a Associação Brasileira das Empresas de Software (ABES), a ES constitui uma área cuja demanda está crescendo significativamente, pois os usuários exigem cada vez mais eficiência, eficácia, dentre outras características de qualidade importantes para um produto tão especial como o software (ABES, 2017). Isto decorre tanto da importância do software em si quanto dos desafios relacionados à formação completa de um profissional que irá atuar no mercado, resultando em uma demanda crescente por profissionais bem qualificados (MORENO et al., 2012).

Em relação à importância do software, nunca as pessoas dependeram tanto de sistemas de informação, pois quase tudo depende de software (MEIRA, 2015). Quanto à formação de profissionais, tipicamente, esses se formam em cursos de graduação na área de Computação (Sistemas de Informação, Ciência da Computação, Engenharia de Software e Engenharia da Computação) a fim de se preparar para atuar na indústria de software (NUNES, REIS e REIS, 2010).

A indústria, entretanto, se queixa de que tais cursos de graduação não ensinam aos estudantes as competências necessárias para que eles possam começar a executar o seu trabalho com eficiência (WANGENHEIM e SILVA, 2009) (MORENO et al., 2012) (MEIRA, 2015). Dessa forma, as empresas de software têm que complementar os conhecimentos dos recém-formados com treinamentos e prover habilidades relacionadas ao processo de desenvolvimento de software (BESSA, CUNHA e FURTADO, 2012), como por exemplo, elicitar requisitos a partir de entrevistas com o cliente.

Esta carência na formação de profissionais graduados na área de ES pode ser resultado de um ensino inadequado das disciplinas da área de ES (LETHBRIDGE et al., 2007). Especialmente nos cursos de graduação, essas disciplinas são normalmente ensinadas de forma bastante superficial (WANGENHEIM e SILVA, 2009) (FOX, 2015). Wangenheim e Silva (2009) realizaram um *survey* com profissionais da área de software que reforçaram as críticas de que as competências de ES, necessárias a estes profissionais, não vêm sendo adequadamente abordadas nessas disciplinas. Segundo Prikladnicki et al. (2009), a maioria dos professores adotam abordagens tradicionais para ensinar ES, como

aulas expositivas, que acabam sendo pouco eficientes, pois são focadas apenas no conteúdo e centradas no professor e não no desenvolvimento de competências.

1.1 MOTIVAÇÃO

Acredita-se que no futuro todo software de uso geral será construído por um engenheiro de software ou por um profissional que tenha conhecimentos específicos em Engenharia de Software (NUNES, REIS e REIS, 2010) (MORENO et al., 2012). Trata-se de uma questão de qualidade e de confiabilidade do produto de software desenvolvido. No entanto, no contexto brasileiro, a indústria de software apresenta escassez de profissionais devidamente qualificados para atuarem em profissões que envolvem as etapas do processo de desenvolvimento de software, compreendidas pela ES (ABES, 2017). Ainda assim, devido à alta demanda por profissionais, a indústria acaba por absorver a maioria destes recém-formados.

Essa escassez de profissionais qualificados pode estar relacionada ao fato de que as competências necessárias não estão sendo adequadamente abordadas nos cursos de graduação (WANGENHEIM e SILVA, 2009). Segundo Wangenheim e Silva (2009), a maioria dos bacharéis de cursos de Computação que trabalham como profissionais da área de software no Brasil tendem a aprender mais sobre tópicos de ES após a graduação. De acordo com Meira (2015, p. 13), a impressão é de que, na área de ES, as graduações foram ultrapassadas pelo conhecimento e pelas práticas dos negócios. As empresas de software investem na capacitação das suas equipes através da realização de cursos e treinamentos para repasse técnico de práticas específicas que compreendem áreas de processo, a fim de desenvolver as competências e as habilidades profissionais necessárias para executar suas atividades. Na Índia, um promissor mercado de software, estima-se que cerca de 80% dos recursos de treinamento das empresas são gastos com profissionais recém-formados (GARG e VARMA, 2008).

Ainda segundo Meira (2015, p. 14), é necessária uma revisão de “como” se criam oportunidades para aprender computação, em especial ES. Não há um consenso em “como” ensinar Engenharia de Software, pois cada universidade adota seus próprios métodos ou abordagens de acordo com a experiência de seus professores (MARQUES, QUISPE e OCHOA, 2014). Do ponto de vista de uma universidade, é uma tarefa desafiadora encontrar novas formas de ensinar os conceitos de ES que proporcione aos estudantes conhecimentos práticos sobre o trabalho que esses deverão executar na

indústria (GNATZ et al., 2003) (MEAD, 2009) (MORENO et al, 2012), pois são muitos os desafios acadêmicos e problemas enfrentados pelos professores, conforme descritos na subseção a seguir.

1.2 DEFINIÇÃO DO PROBLEMA

Como mencionado anteriormente, a indústria apresenta insatisfação quanto ao nível de preparação dos universitários recém-graduados que entram no mercado de trabalho (MEIRA, 2015). Existe um conjunto de razões que pode levar a essa situação. A problemática explorada neste trabalho está relacionada à formação desses profissionais, ou seja, foca-se na abordagem adotada para o ensino de ES durante a graduação. Em relação às atuais abordagens para o ensino de ES, diversos autores relatam problemáticas, como (SOARES, 2008) (CASTRO, GIMENES e MALDONADO, 2008) (SHKOUKANI, 2013) (GIMENES, 2015): (i) muito conteúdo sendo ministrado em pouco tempo; (ii) baixa motivação que os estudantes possuem para estudar os conceitos teóricos de ES; e (iii) dificuldades em preparar os estudantes para a prática profissional dentro de ambientes acadêmicos.

Quanto à problemática (i), o *Curriculum Guidelines* da *Association for Computing Machinery*/*Institute for Electrical and Electronic Engineers* (ACM/IEEE, 2013) e o Currículo de Referência da Sociedade Brasileira de Computação (SBC, 2005) sugerem cerca de 83 tópicos e 125 aprendizagens esperadas para disciplinas da área de ES. Segundo Garg e Varma (2008), a carga horária disponibilizada para disciplinas de ES é insuficiente para discutir essa quantidade de tópicos em profundidade. Se o estudante falta às aulas, o prejuízo é maior ainda. Em geral, os currículos de Computação têm pouco espaço para ES (GIMENES, 2015).

No que diz respeito à problemática (ii), muitos professores ainda optam por abordagens tradicionais, ministrando aulas expositivas e aplicando provas discursivas/objetivas, o que acaba por desmotivar os estudantes que poderiam vim a atuar na área de ES (PRIKLADNICKI et al., 2009) (BESSA, CUNHA e FURTADO, 2012) (FOX, 2015). Neste sentido, o ensino de ES é predominantemente teórico, pois, de acordo com Gimenes (2015), muitos professores preferem apresentar o máximo possível de conteúdo ao invés de trabalhar o desenvolvimento das competências necessárias para formação dos estudantes nessa área.

Por fim, quanto à problemática (iii), os professores não estão conseguindo envolver os estudantes e gerenciar suas expectativas devido às limitações, como escopo e tempo reduzido, inerentes a projetos práticos em disciplinas de ES (SOARES, 2008) (NUNES, REIS e REIS, 2010) (GIMENES, 2015). O protelamento da prática de desenvolvimento de software desmotiva o aprendizado do conteúdo de ES (GIMENES, 2015).

A partir das problemáticas expostas nesta seção, é possível destacar duas lacunas principais:

I. Muitos professores adotam abordagens tradicionais de ensino, ministrando aulas expositivas e aplicando provas discursivas/objetivas. Isso acaba por desmotivar os estudantes e dificultar a sua aprendizagem, tendo em vista que a ES é uma disciplina que possui uma grande base teórica e que a aprendizagem efetiva de seus tópicos se dá principalmente a partir da aplicação prática desses tópicos em projetos (WANGENHEIM e SILVA, 2009) (PRIKLADNICKI et al., 2009) (FOX, 2015);

II. Há diversas limitações no ambiente acadêmico para a realização de projetos práticos de desenvolvimento, como o uso de um problema e cliente fictícios, falta de treinamentos em ferramentas e técnicas específicas, restrições nas decisões que os estudantes podem tomar, dentre outras (SOARES, 2008) (CASTRO, GIMENES e MALDONADO, 2008) (SHKOUKANI, 2013) (GIMENES, 2015).

Existem diversas pesquisas disponíveis na literatura relacionadas a essas problemáticas, conforme apresentam os trabalhos relacionados na Seção 6.2. No entanto, apesar das contribuições relevantes dessas pesquisas, ainda existem algumas lacunas na área de ensino de ES:

1. As abordagens de ensino propostas não adotam estratégias que alterem a atual dinâmica do meio acadêmico. Os engenheiros de software são educados da mesma maneira como têm sido há anos (MARQUES, QUISPE e OCHOA, 2014) (MEIRA, 2015). Segundo Meira (2015, p. 13), os métodos tradicionais de ensino parecem não ter acompanhado a dinâmica das práticas dos negócios, tornando-se pouco efetivos na formação do profissional que a indústria necessita. O uso de abordagens tradicionais, como aula expositiva, acaba sendo

pouco eficiente, pois é centrado apenas no professor e não no envolvimento do estudante em atividades práticas (PRIKLADNICKI et al., 2009);

2. Existe uma grande dificuldade no desenvolvimento de competências profissionais dos alunos no ambiente acadêmico (WANGENHEIM e SILVA, 2009) (GIMENES, 2015). Segundo Lethbridge et al. (2007), essa limitação no desenvolvimento de competências acaba por agravar a carência de profissionais habilitados na área de ES.

Em relação à lacuna 1, Gimenes (2015) destaca que estamos diante de um contexto educacional que questiona drasticamente as formas de ensino-aprendizagem. Nesse sentido, é importante considerar a variação de técnicas de ensino e aprendizado (ACM/IEEE, 2013). No entanto, Fox (2015) enfatiza que a maioria dos professores de ES tendem a ser muito teóricos. Isso acaba por prejudicar o desenvolvimento de competências técnicas nos alunos, pois muitos conceitos da ES são melhores fixados na prática.

Em relação à lacuna 2, Meira (2015) destaca que as empresas capacitam seus profissionais no ritmo das mudanças de práticas do negócio, através de aprendizados explícitos, durante a execução de suas atividades, sem contar as oportunidades de aprendizados implícitos. No entanto, o ambiente acadêmico possui diversas limitações relacionadas ao desenvolvimento de projetos práticos, como escopo e tempo reduzido, falta de experiência do professor em determinados tópicos de ES, dentre outras. Essas limitações acabam por distanciar os projetos de software acadêmicos dos projetos conduzidos na indústria, fazendo com que os alunos não se comprometam de fato com as atividades realizadas em sala de aula. Nesse cenário, os alunos acabam por não vivenciarem as oportunidades de aprendizados e, consequentemente, não desenvolvem as competências técnicas esperadas para um profissional que pretende atuar na área de ES.

1.2.1 Contexto

Existe um consenso entre os pesquisadores da área de que abordagens práticas são as mais indicadas para o ensino de Engenharia de Software (PRIKLADNICKI et al., 2009) (SANTOS et al., 2009) (MALIK e ZAFAR, 2012) (MARQUES, QUISPE e OCHOA, 2014) (SANTOS et al., 2014) (SANTOS, ALEXANDRE e RODRIGUES, 2015). Alguns destes autores destacam que esse ensino deve ser mais centrado no aluno, a fim de aumentar sua motivação, participação, relação com outros alunos e,

consequentemente, sua aprendizagem (PRIKLADNICKI et al., 2009) (MALIK e ZAFAR, 2012) (SANTOS, ALEXANDRE e RODRIGUES, 2015). Outros autores destacam que a realização de projetos práticos permite melhorar a eficácia da aprendizagem, promovendo a habilidade dos alunos trabalharem em equipes para resolverem problemas (SANTOS et al., 2009).

Por outro lado, não há um consenso em "como" ensinar Engenharia de Software, pois cada instituição de ensino adota seus próprios métodos ou abordagens de acordo com a experiência de seus professores (MARQUES, QUISPE e OCHOA, 2014). Nesse contexto, de acordo com o mapeamento realizado por Santos et al. (2014), existem estudos reportando a realização de projetos práticos, uso de jogos, discussão de casos práticos, dentre outras abordagens para ensinar ES.

Inicialmente, a abordagem baseada em projetos práticos era amplamente adotada. Contudo, durante os últimos dez anos, abordagens alternativas vêm sendo propostas e adotadas para o ensino de ES (MARQUES, QUISPE e OCHOA, 2014). Sendo assim, não existe atualmente uma abordagem predominante para conduzir essas experiências práticas (MALIK e ZAFAR, 2012), o que existem são abordagens mais ou menos efetivas, dependendo do contexto de ensino.

A maioria desses estudos destaca a necessidade de proporcionar ao aluno a oportunidade de realizar atividades práticas, mas não menciona especificamente nenhuma abordagem pedagógica para tal. 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 e OCHOA, 2014).

Neste trabalho, optou-se por adotar as abordagens práticas relatadas na literatura que são consideradas humanistas, pois permitem um ensino mais centrado no aluno. De acordo com Prikladnicki et al. (2009) e Malik e Zafar (2012), o uso de jogos e a adoção de práticas da indústria permitem motivar os alunos, aumentar sua participação e interação com a turma, o que acaba por impactar positivamente na sua aprendizagem e, consequentemente, no desenvolvimento de suas competências. De maneira semelhante, Santos, Alexandre e Rodrigues (2015) destacam que a realização de projetos práticos e a adoção de *Problem-Based Learning* (PBL) 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.

Conforme destaca Mizukami (1986), na abordagem humanista, o professor deve atuar como facilitador da aprendizagem, ficando o conteúdo a cargo das escolhas dos alunos e experiências advindas destas. A fim de que, independente destas escolhas, os alunos possam estudar os tópicos recomendados pelos currículos de referência da área, o modelo proposto foi instrumentalizado a partir das unidades de conhecimento da ES definidas pela ACM/IEEE (2013). Assim, os alunos poderão realizar diversas atividades práticas de desenvolvimento de software e, consequentemente, desenvolver competências técnicas relacionadas a estas. Essas competências, por sua vez, foram instrumentalizadas no modelo a partir das atividades técnicas descritas no SWECOM (IEEE COMPUTER SOCIETY, 2014A), conforme destaca a Subseção 2.4.2.

Para que o modelo permitisse com que o aluno planejasse o seu caminho de aprendizagem, a partir de uma experiência concreta e, ao final desta, pudesse refletir e concluir sobre sua aprendizagem, organizaram-se as etapas do modelo de acordo com o ciclo de Kolb (1984). De forma complementar, considerando o estudo de Fleming e Mills (1992), o modelo adotou em cada etapa abordagem que ativam diferentes estímulos sensoriais, permitindo assim atingir alunos com perfis de aprendizagem distintos.

Por fim, para que o aluno pudesse aprender de maneira eficiente os tópicos de ES, ou seja, desenvolver competências técnicas, o modelo estabeleceu níveis incrementais de aprendizagem de acordo com o avanço das fases no ciclo iterativo. Assim, inicialmente o aluno poderá desenvolver a capacidade de observação e recuperação da informação, em seguida entender essa informação e, por fim, aplicar o conhecimento obtido em situações novas e concretas.

1.3 VISÃO GERAL DA PROPOSTA

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): *Se os professores da área de Engenharia de Software (ES) adotarem abordagens de ensino focadas no aluno e práticas de capacitação da indústria, o desenvolvimento de competências técnicas será mais eficiente do que se adotarem abordagens focadas no conteúdo?*

No contexto dessa pesquisa, considera-se “eficiente” o ato de atingir o objetivo esperado (desenvolver competências técnicas em ES) utilizando a menor quantidade de

recursos disponíveis, como a carga horária das disciplinas, o ambiente de sala de aula, os materiais de apoio, a experiência dos professores, dentre outros.

Para responder a esta questão de pesquisa, é necessário entender as atuais abordagens de ensino focadas no aluno descritas na literatura e entender as estratégias de capacitação adotadas pela indústria de software. Dessa forma, o objetivo da pesquisa apresentada nessa tese pode ser definido como:

Objetivo da Pesquisa (OP): *Definir um modelo iterativo baseado nas principais abordagens focadas no aluno que são aplicadas no ensino de Engenharia de Software e que incorpore as práticas de capacitação adotadas pela indústria a fim de que os estudantes desenvolvam competências técnicas em nível de aplicação.*

Por objetivar o desenvolvimento de competências técnicas em ES, esse modelo pode ser aplicado em qualquer curso de graduação da área de Computação (Sistemas de Informação, Ciência da Computação, Engenharia de Software ou Engenharia da Computação), pois não busca, diretamente, atender a um perfil de egresso específico.

Não se caracteriza como objetivo dessa pesquisa criar uma nova abordagem de ensino, mas sim integrar as abordagens práticas identificadas na literatura em um único modelo iterativo a fim de estabelecer um ciclo de aprendizagem (KOLB, 1984) e contemplar os diferentes estilos de aprendizagem dos alunos (FLEMING e MILLS, 1992).

Como diferencial, incorporou-se nesse modelo práticas de capacitação adotadas pela indústria de software, como *coaching* e *mentoring* (GOMES et al., 2015). Essas práticas permitem aos profissionais-alvos da aplicação tanto a aquisição de novas competências quanto o desenvolvimento das já existentes. Dessa maneira, pretende-se que os alunos possam desenvolver competências técnicas em nível de aplicação de conhecimento (Subseção 2.4.4), de forma similar ao nível requerido para um profissional iniciante na área de Engenharia de Software (IEEE COMPUTER SOCIETY, 2014A).

Dadas as diversas limitações do ambiente acadêmico, pretende-se adaptar essas práticas de acordo com os recursos disponíveis nesse ambiente. Por exemplo, os professores podem realizar *mentoring* no projeto prático das disciplinas da área de ES, orientando os estudantes em suas escolhas, apresentando as vantagens e desvantagens de determinadas técnicas e métodos.

1.3.1 Justificativa

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). 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 (ACM/IEEE, 2013). Além de tratar o conteúdo, se faz necessário que os educadores tratem dos aspectos didáticos e pedagógicos no ensino (ENRICONE, 2002). Nesse contexto, destaca-se que a realização de aulas tradicionais, permeadas por estratégias de ensino focadas no conteúdo e centradas no professor, raramente satisfazem a necessidade de aprendizagem dos estudantes (GRILLO, 2003) (PRIKLADNICKI et al., 2009).

No atual cenário, os recursos e os esforços investidos no ensino dos tópicos e no desenvolvimento de competências e habilidades em ES durante a graduação se perdem em quase sua totalidade. A instanciação dos currículos de graduação em termos de tópicos, ao invés da sua unificação ao redor de problemas que os profissionais irão enfrentar na prática, acaba por criar um cenário que raramente dará resultados. Por causa disso, são as empresas que têm, na prática, que investir para formar seu capital humano de acordo com suas necessidades (MEIRA, 2015).

1.3.2 Relevância

A proposta apresentada nesse trabalho assume relevância dada a carência de profissionais qualificados na área de ES. Ao buscar tratar essa carência na formação profissional, a relevância desse trabalho se estende para o meio acadêmico por consequência. Essa relevância se apresenta à medida que o trabalho pretende colaborar com o desenvolvimento de competências profissionais durante a graduação a partir de adequação de estratégias de capacitação da indústria para o ambiente acadêmico.

No contexto da indústria, destaca-se que a qualidade da educação é um dos fatores importantes que influenciam na qualidade dos profissionais, e uma das maneiras de se

aumentar a qualidade do ensino é aperfeiçoar a didática e estratégias dos processos de ensino e aprendizagem (SAVI, 2011). A relevância dessa pesquisa se soma, de maneira indireta, aos esforços das empresas brasileiras em desenvolver produtos de software com a qualidade esperada pelos usuários e assim tornarem-se competitivas no mercado nacional e internacional.

No contexto acadêmico, destaca-se a relevância desse trabalho dada a aplicabilidade da proposta. A maioria dos currículos de graduação foca no conteúdo e não na aplicação deste em problemas e projetos que o graduando enfrentará na prática (MORENO et al., 2012) (MEIRA, 2015). De acordo com Fox (2015), a adoção de abordagens tradicionais de ensino faz com que os alunos releguem as práticas de ES e continuem desenvolvendo software da mesma forma que estão habituados.

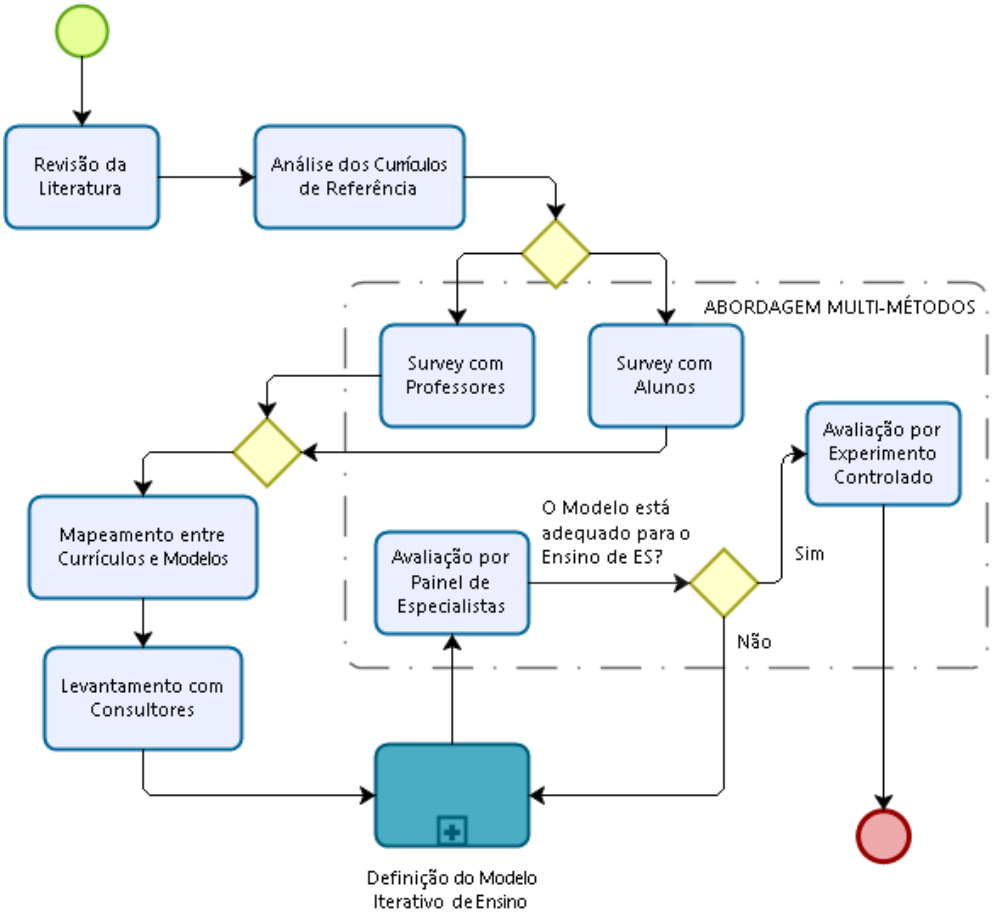
O modelo proposto nesta tese visa auxiliar os professores na adoção de abordagens práticas que permitem lecionar esse conteúdo a partir da sua aplicação. Dessa forma, espera-se que o professor possa alterar a atual dinâmica de ensino de ES e que o aluno possa desenvolver as competências técnicas requeridas pela indústria.

1.4 METODOLOGIA DE PESQUISA

O desenho de pesquisa dessa tese é baseado em uma abordagem multi-métodos que combina dois ou mais métodos quantitativos e/ou qualitativos em um único estudo (HESSE-BIBER, 2010), como nesta pesquisa, um *survey*, um painel de especialistas e um experimento controlado. Nessa abordagem, a triangulação é usada para consolidar os resultados para os diferentes métodos, considerando que as mesmas questões de pesquisa serão investigadas em cada um desses métodos. Dessa forma, a coleta de dados de várias fontes diferentes ajuda a reforçar a validade do estudo (EASTERBROOK et al., 2007).

Na abordagem multi-métodos, as principais decisões envolvem a estratégia para coleta de dados e a definição da sequência em que são empregados métodos diferentes. A **Figura 1** mostra como a metodologia de pesquisa foi aplicada nessa tese. O desenho de pesquisa inicia-se a partir da definição do objetivo e da questão de pesquisa da proposta de tese, seguida da realização de uma revisão da literatura. Essa revisão permitiu identificar as principais definições e conceitos da área de ensino de Engenharia de Software e contribuiu para a fundamentação teórica. Permitiu também identificar princípios e abordagens para o ensino de ES em trabalhos relacionados a essa pesquisa, além de outras abordagens que dão um enfoque alternativo à proposta apresentada.

Figura 1 – Metodologia de Pesquisa da Tese



Fonte: Elaborado pelo autor.

Em seguida, realizou-se uma análise dos currículos de referência da ACM/IEEE (2013) e da SBC (2005) a fim de identificar os tópicos de ES recomendados por esses e as competências técnicas que os estudantes devem desenvolver durante a realização dessa disciplina. Então, um *survey* foi conduzido com professores a fim de identificar quais desses tópicos são adotados nas ementas das disciplinas de ES e quais abordagens de ensino e avaliação são aplicadas por esses. Paralelamente, conduziu-se um *survey* com estudantes concluintes das disciplinas da área de ES a fim de analisar o grau de aprendizado em relação a esses tópicos, além das suas preferências em relação às abordagens de ensino e avaliação adotadas pelos professores. Esses *surveys* permitiram identificar os tópicos de ES mais adotados pelos professores e as abordagens de ensino e avaliação consideradas mais efetivas pelos estudantes. Além disso, permitiram reforçar as problemáticas identificadas na literatura especializada.

Com o intuito de relacionar os tópicos mais adotados pelos professores com as melhores práticas da indústria de software, realizou-se um mapeamento entre esses

tópicos e as recomendações dos modelos de qualidade (SEI, 2010) (SOFTEX, 2012). A partir desse mapeamento, consultores em MPS, que também atuam como professores de ES, responderam um questionário a fim de levantar as estratégias de capacitação adotadas por estes na indústria para desenvolver competências profissionais.

Após a identificação das práticas de capacitação adotadas pela indústria e das abordagens de ensino identificadas na literatura e adotadas pelos professores, definiu-se o modelo de ensino proposto nesta tese. Esse modelo é composto por uma sintaxe que define as etapas do ciclo iterativo; um sistema de apoio que especifica os requisitos e materiais necessários para aplicar o modelo em sala de aula; e os efeitos esperados nos estudantes a partir da aplicação desse modelo no ensino de ES.

Por fim, realizou-se a avaliação da proposta em duas etapas. Na primeira etapa, avaliou-se a documentação e usabilidade do modelo através de um painel de especialistas, composto por 3 (três) doutores que atuam como professores/pesquisadores na área de ES. Após a atualização do modelo, mediante o parecer conclusivo do painel de especialistas que considerou o modelo adequado para o ensino de ES, buscou-se avaliar a instanciação deste por professores em sala de aula. Sendo assim, nessa segunda etapa, realizou-se um experimento controlado em 2 disciplinas da área de ES. Nesse experimento, o grupo de controle seguiu uma abordagem de ensino tradicional focada no professor. Já um dos grupos experimentais seguiu uma abordagem de ensino iterativa focada no aluno e o outro seguiu essas mesmas abordagens somadas às práticas de capacitação da indústria. O objetivo foi comparar o aprendizado dos estudantes desses três grupos e, consequentemente, o desenvolvimento de competências técnicas em ES. Adicionalmente, os professores que usaram o modelo proposto avaliaram a sua usabilidade.

É importante destacar que, a fim de realizar a triangulação dos resultados, o questionário usado no *survey* com os alunos e os critérios de avaliação da usabilidade usados no painel de especialistas foram reutilizados no experimento controlado. Assim, pôde-se comparar e reforçar os resultados obtidos na execução desses métodos.

1.5 FORA DO ESCOPO

Como esse trabalho está inserido em uma área de conhecimento bastante ampla, um conjunto de aspectos relacionados estará fora do escopo da pesquisa. Sendo assim, os seguintes tópicos não serão diretamente tratados nesta tese:

1. **Definição dos conteúdos de ensino.** O modelo apresentado nessa tese segue a abordagem humanista (ver Subseção 2.3.2), onde o conteúdo é considerado como secundário no processo de ensino-aprendizagem. Dessa forma, não é pretensão desse trabalho especificar o conteúdo das disciplinas de ES.
2. **Avaliação da aprendizagem dos alunos.** O modelo de ensino proposto não especifica técnicas de avaliação de aprendizagem, pois, de acordo com Joyce, Weil e Calhoun (2015), a avaliação não faz parte diretamente dos componentes de um modelo (ver Seção 2.2). No entanto, Mizukami (1986) argumenta que, na abordagem humanista, essas técnicas devem ser discutidas e selecionadas através do consenso entre professores e alunos.
3. **Capacitação de Professores em Práticas da Indústria.** A abordagem proposta nessa tese não cobre a capacitação de professores em práticas da indústria. Muitos professores não possuem vivência prática de mercado. Isto pode dificultar a adoção do modelo por parte desses. No entanto, a proposta é baseada em práticas recomendadas por consultores que também atuam como professores, a fim de contemplar as experiências de docência desses. Além disso, uma das contribuições da proposta é a adaptação das práticas do mercado para o contexto acadêmico. Assim, espera-se que o professor, mesmo que não tenha prática de mercado, possa adotar o modelo proposto.

1.6 ESTRUTURA DO TRABALHO

Este capítulo introdutório apresenta a motivação, define o problema de pesquisa e descreve uma visão geral da proposta de solução desse problema, através dos seus objetivos, justificativa e relevância. Por fim, descreve a metodologia seguida para atendimento dos objetivos definidos, os assuntos que ficaram de fora do escopo e a estrutura desta tese.

O Capítulo 2 apresenta as definições de modelos de ensino e discorre sobre as abordagens de ensino-aprendizagem que são tratadas nessa tese: Tradicional e Humanista. Posteriormente, esse capítulo mostra os principais conceitos relacionados ao ensino-aprendizagem de Engenharia de Software e descreve as práticas de capacitação da indústria incorporadas nesse modelo.

O Capítulo 3 apresenta os métodos de pesquisa seguidos durante a realização desse trabalho. Inicialmente, detalha o *framework* metodológico e os principais tipos de validade dos métodos empíricos. Em seguida, descreve o desenho de pesquisa através do *design*, participantes, resultados, análise e discussões e ameaças à validade dos resultados de cada um dos métodos de pesquisa.

O Capítulo 4 apresenta a principal contribuição dessa pesquisa, o modelo iterativo para o ensino de Engenharia de Software. Inicialmente, descreve os princípios e as abordagens práticas adotadas para o ensino de ES. Em seguida, descreve a sintaxe do modelo proposto a partir das suas 6 (seis) etapas. Discorre, ainda, sobre a interação entre os agentes do processo de ensino-aprendizagem, através da descrição do sistema social do modelo. Por fim, especifica os requisitos e materiais de apoio necessários para aplicação do modelo.

O Capítulo 5 detalha como se deu a avaliação do modelo através de um painel de especialistas e da execução de um experimento controlado em 2 fases. Assim, descreve o planejamento dessas avaliações através do *design* e participantes, bem como apresenta os resultados obtidos, realizando uma análise quantitativa e qualitativa. Adicionalmente, discorre sobre como as ameaças à validade foram tratadas. Por fim, apresenta a triangulação dos resultados do experimento controlado com o painel de especialista e com o *survey* e discute as principais lições aprendidas nesta avaliação.

O Capítulo 6 é dedicado à apresentação das discussões sobre os resultados obtidos e suas implicações. Assim, inicialmente, apresenta os trabalhos relacionados que podem ser adotados como uma alternativa ao modelo proposto. Em seguida, discute a relação do modelo com esses trabalhos relacionados através de uma análise comparativa. Por fim, discorre sobre as implicações desta pesquisa tanto para a academia quanto para a indústria e as principais lições aprendidas durante a execução deste trabalho.

Finalmente, o Capítulo 7 apresenta as principais conclusões obtidas a partir da realização desse trabalho. Em seguida, descreve as principais contribuições dessa pesquisa e uma análise das limitações dessas. Por fim, são apresentados os trabalhos futuros dessa tese de doutorado, que visam reforçar as conclusões obtidas e aumentar o escopo dessa pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 INTRODUÇÃO

O termo Engenharia de Software (ES) foi introduzido no final da década de 60 como uma resposta à chamada crise do software, quando não se realizavam, de maneira adequada, o planejamento, a comunicação e a documentação, resultando na falha de diversos projetos e, consequentemente, em inúmeros casos de prejuízos financeiros (BOEHM, 1976). Nesse cenário, a ES foi definida a fim de estabelecer e utilizar sólidos princípios de engenharia para produzir software de forma mais econômica, eficiente e confiável (NUNES, 2015). Desde então, a ES passou a ser presença constante na academia e na indústria.

No contexto acadêmico, a Engenharia de Software é uma área dedicada à aplicação de teoria, conhecimento e prática para o desenvolvimento efetivo e eficiente de sistemas de software que satisfaçam aos requisitos dos usuários (ACM/IEEE, 2013). Tradicionalmente, as disciplinas da área de ES são tratadas no ensino formal no nível de graduação e/ou pós-graduação ou por meio de treinamentos profissionais de curta duração (SANTOS et al., 2014). Esta tese foca no nível de graduação, onde diversas teorias discutem a forma como essa área deve ser abordada por professores.

De maneira geral, os cursos de bacharelado da área de Computação como Engenharia da Computação, Sistemas de Informação, Ciência da Computação e, mais recentemente, Engenharia de Software, objetivam desenvolver nos estudantes a capacidade para aplicar seus conhecimentos de forma independente e desenvolver competências e habilidades específicas, como trabalho em grupo, comunicação escrita e verbal e resolução de problemas (SBC, 2005) (ACM/IEEE, 2013). Para tanto, devem adotar abordagens de ensino que contemplem as competências a serem desenvolvidas, as atividades práticas e laboratoriais, as metodologias de ensino-aprendizagem (SBC, 2005), além de considerar os perfis dos alunos.

Este capítulo tem como objetivo apresentar os principais construtos envolvidos nessa pesquisa através das suas definições e conceitos. Sendo assim, apresentam-se as definições de modelos de ensino, bem como as classificações desses de acordo com o processo de ensino-aprendizagem. Além disso, descrevem-se os principais conceitos relacionados ao ensino-aprendizagem de ES, as unidades de conhecimentos que

compõem essa área, as competências técnicas a serem desenvolvidas, os diferentes estilos e níveis de aprendizagem. Por fim, destacam-se as práticas de capacitação mais adotadas pela indústria de software para desenvolver competências técnicas em ES nos profissionais.

2.2 MODELOS DE ENSINO

Existem diversas definições na literatura para o termo “modelo de ensino”. Uma das mais aceitas e referenciadas é a dos autores Joyce e Weil (1972), que define modelo de ensino como “um padrão ou plano que pode guiar a definição de currículos ou cursos, auxiliar a seleção de materiais de instrução e orientar as ações de um professor”.

Nesse sentido, esses autores enfatizam que modelos de ensino são apenas instrutivos, consistindo em orientações para a concepção de atividades e ambientes educacionais. Assim, especificam formas de ensino e aprendizagem que se destinam a atingir determinados tipos de metas (JOYCE e WEIL, 1972).

Adicionalmente, Passi, Singh e Sansanwal (1991) destacam que o modelo de ensino pode ser definido como um *design* instrucional que trata de um conjunto de técnicas, métodos e recursos que podem ser utilizados em um processo de aprendizagem.

A partir dessas definições, é possível destacar as principais funções de um modelo de ensino (JOYCE, WEIL e CALHOUN, 2015):

- Orientar o professor na seleção de técnicas, estratégias e métodos adequados para o processo de ensino e na seleção do material usado para o atendimento dos objetivos de ensino;
- Provocar mudanças desejáveis no comportamento dos alunos;
- Propor formas e meios de criar uma situação ambiental favorável para a realização do processo de ensino;
- Atingir o nível desejável de interação professor-aluno durante o processo de ensino;
- Ajudar na seleção adequada do material de instrução para o ensino do curso ou do currículo planejado;
- Auxiliar na elaboração de atividades educacionais apropriadas;
- Ajudar no procedimento de criação de materiais que sejam fontes de aprendizagem interessantes e eficazes;

- Estimular o desenvolvimento de inovações educacionais;
- Contribuir na formação da teoria do ensino;
- Ajudar a estabelecer a relação de ensino e aprendizagem empiricamente.

Baseado nessas funções, Pateliya (2013) elenca as principais características que todo modelo de ensino deve contemplar:

- **Resultados de aprendizagem:** descrição do que os alunos devem realizar após completar uma sequência instrucional;
- **Meio ambiente:** descrição da condição ambiental em que a resposta de um aluno deve ser observada;
- **Critérios de desempenho:** descrição dos critérios de desempenho que se esperam dos alunos;
- **Operação:** descrição dos mecanismos que fornecem respostas para a reação dos estudantes e para a interação com o meio ambiente;
- **Processo Científico:** embasamento do modelo de ensino em um procedimento sistemático para modificar o comportamento do aluno.

A partir dessas características, é possível destacar os componentes que todo modelo de ensino deve especificar. Joyce, Weil e Calhoun (2015) examinaram diversos modelos de ensino, desde a primeira edição do seu livro intitulado *Models of Teaching* (JOYCE e WEIL, 1972), com o objetivo de categorizar os componentes individuais desses modelos. Como resultado, identificaram 6 (seis) componentes principais:

1. **Foco:** é a intenção central do modelo, o objetivo do ensino. Os elementos da sintaxe giram em torno do objetivo principal do modelo;
2. **Sintaxe:** descreve a estrutura do modelo através dos elementos de ensino. Inclui a sequência de passos envolvidos na organização do modelo;
3. **Sistema Social:** descreve a relação entre o professor e os alunos, bem como o papel desempenhado por cada um nas atividades que definem a natureza do sistema social. Diz respeito às funções interativas e relações entre o professor e o aluno, às normas esperadas e quais os comportamentos dos alunos devem ser recompensados. As táticas e estratégias motivacionais para envolver os alunos também devem ser discutidas neste componente;

4. **Princípios de Reação:** especificam ao professor como lidar com o aluno e como reagir às respostas destes durante o uso do modelo. As respostas na utilização de um modelo designado devem ser apropriadas e coerentes com seu foco. Através desses princípios que o professor percebe se os alunos têm se envolvido ativamente nas etapas do modelo;
5. **Sistema de Apoio:** define as condições de apoio necessárias para implementar o modelo com sucesso. Refere-se a quaisquer requisitos adicionais necessários à implementação do modelo, para além daquelas habilidades e capacidades geralmente possuídas por professores. Esse apoio também inclui requisitos de materiais de referência, permissões, instalações de equipamentos, laboratórios ou ambiente de aprendizagem que precisam ser acessados na utilização do modelo;
6. **Aplicação e Efeitos:** especifica como os alunos podem aplicar o que o modelo ensina. Essa aplicação diz respeito à utilidade do modelo, uma vez que pode ser transferida para outras situações e experiências.

O modelo de ensino derivado desta pesquisa adota as definições de Joyce e Weil (1972), bem como incorpora as principais características descritas por Pateliya (2013). Adicionalmente, esse modelo é especificado a partir da descrição dos componentes identificados por Joyce, Weil e Calhoun (2015), que derivam a estrutura da sua documentação.

2.3 ABORDAGENS DE ENSINO-APRENDIZAGEM

O processo de ensino-aprendizagem tem sido estudado segundo diferentes enfoques, muitos deles relacionados com o momento histórico de sua criação e do desenvolvimento da sociedade na qual estavam inseridos. De acordo com Santos (2005), esse processo de ensino-aprendizagem é composto de duas partes: ensinar, que exprime uma atividade, e aprender, que envolve certo grau de realização de uma determinada tarefa com êxito.

Diversos autores analisaram as abordagens do processo de ensino-aprendizagem existentes a partir de seus princípios, dos componentes necessários ao fenômeno educativo e de seus efeitos sobre o indivíduo e a sociedade. Dentre esses, destaca-se o trabalho de Mizukami (1986) que considera que a base das teorias do conhecimento

envolve três características básicas: o sujeito, o objeto de estudo e a interação sujeito-objeto. De acordo com essas características, a autora nomeia as diferentes abordagens do processo de ensino-aprendizagem:

- Abordagem Tradicional;
- Abordagem Comportamentalista;
- Abordagem Humanista;
- Abordagem Cognitivista;
- Abordagem Sociocultural.

De acordo com Mizukami (1986), na Abordagem Tradicional o ensino deve ser centrado no professor e o aluno apenas executa prescrições que lhe são fixadas por esse. Na Abordagem Comportamentalista, o professor possui a responsabilidade de planejar e desenvolver o sistema de ensino-aprendizagem, de forma tal que o desempenho do aluno seja maximizado. Assim, os comportamentalistas consideram a experiência ou a experimentação planejada como a base do conhecimento.

Já na Abordagem Humanista, o professor em si não transmite o conteúdo, atuando como facilitador da aprendizagem. Nessa abordagem, o conteúdo advém das próprias experiências do aluno. Ainda segundo Mizukami (1986), a Abordagem Cognitivista considera o conhecimento como uma construção contínua. Neste sentido, cabe ao professor criar situações, propiciando condições onde possam se estabelecer reciprocidade intelectual e cooperação com os alunos.

Por fim, a Abordagem Sociocultural define que a elaboração e o desenvolvimento do conhecimento estão ligados ao processo de conscientização. Os professores que adotam essa abordagem devem estar empenhados na prática transformadora, buscando desmitificar e questionar, juntamente com o aluno.

2.3.1 Abordagem Tradicional

A partir da revisão da literatura realizada nesse trabalho, observa-se que a Abordagem Tradicional continua sendo amplamente adotada no ensino de ES (PRIKLADNICKI et al., 2009) (BESSA, CUNHA e FURTADO, 2012) (FOX, 2015). Entende-se por Abordagem Tradicional a prática educativa caracterizada pela transmissão dos conhecimentos acumulados pelo professor, agindo independentemente dos interesses dos alunos em relação aos conteúdos das disciplinas (SANTOS, 2005).

Na Abordagem Tradicional, o enfoque é o professor e o seu conhecimento. Nesse contexto, de acordo com Santos (2005), ocorre a denominada “pedagogia da transmissão”, que valoriza sobretudo os conteúdos educativos e tende a formar “alunos passivos” que são vistos como simples “depositário” do conhecimento do professor.

Segundo Mizukami (1986), nessa abordagem, a escola (ou instituição de ensino) é o lugar onde se realiza a educação, a qual se restringe a um processo de transmissão de informações em sala de aula. As possibilidades de cooperação entre pares são reduzidas, posto que a natureza de grande parte das atividades destinadas aos alunos exige participação individual de cada um deles.

A partir da descrição dessas principais características, pode-se identificar na **Tabela 1** os elementos relevantes sobre a Abordagem Tradicional.

Tabela 1 – Elementos da Abordagem Tradicional

Elemento	**Descrição**
A escola	Deve ser organizada com funções claramente definidas. Também deve definir normas disciplinares rígidas e preparar os indivíduos para a sociedade.
O aluno	Um ser “passivo” que deve assimilar os conteúdos transmitidos pelo professor.
O professor	É o transmissor dos conteúdos aos alunos. Deve predominar como autoridade.
Ensino e Aprendizagem	Os objetivos educacionais obedecem à sequência lógica dos conteúdos. Predominam aulas expositivas, com exercícios de fixação e leitura de apostilas.

Fonte: Adaptado de Santos (2005).

A relação professor-aluno é vertical, sendo que o professor detém o poder decisório quanto à metodologia, ao conteúdo, à avaliação, à forma de interação na aula e afins. A ênfase é dada às situações de sala de aula, onde os alunos são "instruídos" e "ensinados" pelo professor (MIZUKAMI, 1986). Por fim, Mizukami (1986) enfatiza que uma das consequências do ensino tradicional, já que a aprendizagem consiste em aquisição de informações e demonstrações transmitidas, é a formação de reações estereotipadas e aplicáveis, quase sempre, somente às situações idênticas em que foram adquiridas. O aluno que "aprendeu" segundo essa abordagem apresenta, com frequência, compreensão apenas parcial.

2.3.2 Abordagem Humanista

Na Abordagem Humanista, o enfoque é o sujeito, com o "ensino centrado no aluno". Assim, o professor deve ser um "facilitador da aprendizagem", ou seja, deve fornecer condições para que os alunos aprendam, podendo ser treinado para tomar atitudes favoráveis condizentes com essa função (SANTOS, 2005). Nessa abordagem, os conteúdos de ensino são vistos como externos e assumem papel secundário, privilegiando-se o relacionamento das pessoas envolvidas no processo de ensino e aprendizagem (MIZUKAMI, 1986).

A partir desses pressupostos, pode-se identificar na **Tabela 2** os elementos relevantes sobre a Abordagem Humanista.

Tabela 2 – Elementos da Abordagem Humanista

Elemento	**Descrição**
A escola	Deve oferecer condições ao desenvolvimento e autonomia do aluno.
O aluno	Um ser "ativo". Centro do processo de ensino e aprendizagem. Aluno criativo, que "aprendeu a aprender". Aluno participativo.
O professor	É o facilitador da aprendizagem.
Ensino e Aprendizagem	Os objetivos educacionais obedecem ao desenvolvimento psicológico do aluno. Os conteúdos programáticos são selecionados a partir dos interesses dos estudantes. A avaliação valoriza aspectos afetivos (atitudes), com ênfase na autoavaliação.

Fonte: Adaptado de Santos (2005).

Considerando os elementos do processo de ensino-aprendizagem, observou-se que a Abordagem Humanista é predominante no modelo apresentado nesse trabalho. Essa predominância se dá em função do foco do modelo e dos papéis a serem assumidos pelos professores e alunos durante a adoção desse em sala de aula.

De acordo com Mizukami (1986), a escola (ou instituição de ensino) tem como finalidade primeira a criação de condições que facilitem a aprendizagem do aluno de forma que seja possível seu desenvolvimento tanto intelectual quanto emocional. Assim, deve criar condições nas quais os alunos possam tornar-se pessoas de iniciativas, de

responsabilidade, autodeterminação que saibam aplicar a aprendizagem no que lhe servirão de solução para seus problemas.

Devido à complexidade e às diferenças entre as diversas instituições de ensino no Brasil, o elemento "escola" não será diretamente tratado por essa pesquisa. Sendo assim, as condições que facilitam a aprendizagem do aluno deverão partir do professor. Quanto às motivações e iniciativas, deverão partir dos próprios alunos.

No que diz respeito ao processo de ensino e aprendizagem, Mizukami (1986) destaca que esse dependerá do caráter individual do professor e de como ele se relaciona com o caráter pessoal do aluno. O professor deve assumir a função de facilitador da aprendizagem e nesse clima entrará em contato com problemas vitais que tenham repercussão na existência do estudante. Já o aluno deve responsabilizar-se pelos objetivos referentes à aprendizagem que tenha significado para ele.

No entanto, a Abordagem Humanista não enfatiza técnicas ou métodos para facilitar a aprendizagem. De acordo com Mizukami (1986), cada educador deve elaborar a sua forma de facilitar a aprendizagem no que se refere ao que ocorre em sala de aula. Uma das contribuições do modelo de ensino proposto nessa tese é a sugestão de práticas de capacitação da indústria que proporcionam um aumento no engajamento e na motivação dos alunos (ver Seção 2.5).

Por fim, foi realizada uma adaptação na adoção da Abordagem Humanista para o modelo de ensino no que diz respeito à avaliação. A abordagem considera que só o indivíduo pode conhecer realmente a sua experiência e, portanto, as avaliações devem ocorrer de acordo com padrões prefixados, por autoavaliação dos alunos. Apesar disso, o modelo de ensino não especifica técnicas de avaliação de aprendizagem, pois a avaliação não faz parte diretamente dos componentes de um modelo (ver Seção 1.5).

Sendo assim, a recomendação dos autores deste trabalho é que essas técnicas sejam discutidas e selecionadas através de consenso entre professores e alunos usuários do modelo.

2.4 ENSINO-APRENDIZAGEM DE ENGENHARIA DE SOFTWARE

Existem órgãos representativos na área da Computação, como a Sociedade Brasileira da Computação (SBC) e os internacionais *Association for Computing Machinery* (ACM) e *Institute for Electrical and Electronic Engineers* (IEEE), que

discutem e elaboram currículos de referência a fim de especificar diretrizes de ensino e formação profissional nas áreas que abrangem a Computação, sendo uma delas a Engenharia de Software. Assim, os currículos de referência da ACM/IEEE (2013) e SBC (2005) recomendam que os cursos de graduação contemplem os tópicos das unidades de conhecimento da Engenharia de Software que permitam desenvolver as competências esperadas para os profissionais da área. Por exemplo, pode-se aplicar os tópicos da unidade de Gerenciamento de Projeto a fim de desenvolver nos graduandos a competência de trabalhar em grupo, e, para desenvolver a habilidade de comunicação escrita e verbal, pode-se utilizar das técnicas relacionadas à unidade de Engenharia de Requisitos.

De acordo com Nunes (2015), faz-se necessário identificar em primeiro lugar quais as competências profissionais a serem desenvolvidas para então encontrar os conteúdos que habilitem os alunos a desenvolverem essas competências. Adicionalmente, busca-se entender como usar as abordagens de ensino eficazmente para contribuir com a formação e com o aumento da competência dos alunos (SANTOS et al., 2014). Para tal, deve-se analisar quais abordagens de ensino são mais adequadas para os diferentes perfis de alunos. É necessário, assim, identificar e conhecer os diferentes estilos e níveis de aprendizagem.

As subseções a seguir tratam das unidades de conhecimento da Engenharia de Software consideradas nesse trabalho, bem como o referencial teórico para a análise e avaliação das competências técnicas em ES. Adicionalmente, abordam as diferentes teorias de estilos de aprendizagem e a classificação da aprendizagem em níveis.

2.4.1 Unidades de Conhecimento

A SBC classifica a área de Engenharia de Software como pertencente ao grupo das Tecnologias da Computação, o qual “compreende o núcleo de matérias que representam um conjunto de conhecimento agregado e consolidado que capacitam o aluno para a elaboração de solução de problemas nos diversos domínios de aplicação” (SBC, 2005).

Segundo a SBC (2005), são tópicos relacionados com o ensino de Engenharia de Software: Processo de Desenvolvimento de Software; Ciclo de Vida de Desenvolvimento de Software; Qualidade de Software; Técnicas de Planejamento e Gerenciamento de Software; Gerenciamento de Configuração de Software; Engenharia de Requisitos; Métodos de Análise e de Projeto de Software; Garantia de Qualidade de Software;

Verificação, Validação e Teste; Manutenção; Documentação; Padrões de Desenvolvimento; Reuso; Engenharia Reversa; Reengenharia; Ambientes de Desenvolvimento de Software.

Já a ACM/IEEE classifica a Engenharia de Software como uma das Áreas de Conhecimento (*Knowledge Areas*) do seu Corpo de Conhecimento (*Body of Knowledge*). Cada área corresponde a um tópico de estudo da Computação, sendo essas, por sua vez, organizadas em um conjunto de Unidades de Conhecimento (*Knowledge Units*) (ACM/IEEE, 2013).

Segundo a ACM/IEEE (2013), são unidades de conhecimento relacionadas com o ensino de Engenharia de Software: Processo de Software; Gerenciamento de Projeto de Software; Ferramentas e Ambientes; Engenharia de Requisitos; Projeto de Software; Desenvolvimento de Software; Verificação e Validação; Evolução de Software; Confiabilidade de Software; Métodos Formais. Cada uma dessas unidades é composta por um conjunto de Tópicos relacionados às aprendizagens esperadas.

A partir da análise desses currículos de referência, observa-se a possibilidade de correlacionar as unidades de conhecimento da ACM/IEEE aos principais tópicos da SBC, conforme apresenta o mapeamento realizado pelos autores na **Tabela 3**.

Tabela 3 – Mapeamento entre o Currículo de Referência da ACM/IEEE e SBC

Unidades de Conhecimento da ACM/IEEE (2013)	**Tópicos da SBC (2005)**
Processo de Software	Processo de Desenvolvimento de Software Ciclo de Vida de Desenvolvimento de Software
Gerenciamento de Projeto de Software	Técnicas de Planejamento e Gerenciamento de Software Garantia de Qualidade de Software Documentação
Ferramentas e Ambientes	Gerenciamento de Configuração de Software Verificação, Validação e Teste Ambientes de Desenvolvimento de Software
Engenharia de Requisitos	Engenharia de Requisitos
Projeto de Software	Métodos de Análise e de Projeto de Software Padrões de Desenvolvimento Qualidade de Software
Desenvolvimento de Software	Padrões de Desenvolvimento
Verificação e Validação	Verificação, Validação e Teste
Evolução de Software	Reuso

	Reengenharia Engenharia Reversa Gerenciamento de Configuração de Software
Confiabilidade de Software	Métodos de Análise e de Projeto de Software
Métodos Formais	Não possui correspondente

Fonte: Elaborado pelo autor.

De acordo com Nunes, Reis e Reis (2010), devido à grande quantidade de assuntos que compõem o ensino de ES e à baixa disponibilidade de carga horária para lecioná-los, é necessário realizar uma priorização dos tópicos a serem abordados, em razão do perfil do profissional que se deseja formar. No presente trabalho, optou-se por adotar como base a classificação da ACM/IEEE por diversos motivos.

Primeiramente, o currículo de referência da ACM/IEEE (2013) é reconhecido e adotado internacionalmente. Adicionalmente, o nível de detalhamento do currículo de referência da ACM/IEEE (2013) é maior, pois descreve os tópicos e resultados de aprendizagens de cada unidade de conhecimento enquanto o da SBC apenas descreve os tópicos. Assim, devido o objetivo do modelo de apoiar o ensino de tópicos da ES em disciplinas de qualquer curso de graduação da área de Computação, optou-se por adotar como referência as diretrizes curriculares da ACM/IEEE (2013) para cursos de graduação em Ciência da Computação, visto que esse oferece uma formação mais ampla, integrando diversas áreas da Computação.

2.4.2 Competências Técnicas

As Diretrizes Curriculares Nacionais (DCNs) para os cursos de graduação em Computação (MEC, 2012) sugerem selecionar os conteúdos a serem ministrados com base em refinamentos de habilidades e competências. A partir dessas habilidades e competências, deve-se encontrar conteúdos que habilitem os alunos a desenvolverem estas últimas (NUNES, 2015). Assim, deve-se fazer um questionamento inicial a respeito de quais competências o aluno deve desenvolver.

Há diversos estudos relacionados ao tema desenvolvimento de competências (SAVI, 2011). Magalhães, Wanderley e Rocha (1997) constataram que o termo competência é usado para indicar os atributos necessários que credenciam uma pessoa a exercer determinada função em uma determinada área. Essa definição é semelhante à apresentada por Brandão e Borges-Andrade (2007), que investigaram a noção do termo

no mercado, onde competência é usada para qualificar a pessoa capaz de desempenhar eficientemente determinado papel.

Embora existam diversas pesquisas e definições para conceituar competência, a maioria dos autores converge ao apontar que esse termo está relacionado a três outros termos interdependentes: conhecimento, habilidade e atitude (BRANDÃO e GUIMARÃES, 2001). De acordo com essa visão, conhecimentos, habilidades e atitudes podem ser considerados recursos (ou insumos) para a competência e é por meio dessas três dimensões que ocorre o desenvolvimento de competências (SAVI, 2011).

A dimensão do conhecimento se refere a um conjunto de informações armazenadas na memória da pessoa; a dimensão da habilidade se refere à capacidade de se fazer uso produtivo do conhecimento, saber como fazer algo e; a dimensão da atitude se refere à predisposição da pessoa em relação ao trabalho, a objetos ou a situações (ZABALA e ARNAU, 2010) (BRANDÃO e BORGES-ANDRADE, 2007).

Sendo assim, esse trabalho adota a definição sugerida por Nunes, Yamaguti e Nunes (2016), que define o termo competência como a "capacidade de articular e mobilizar conhecimentos, habilidades e atitudes, colocando-os em ação para resolver problemas e enfrentar situações de imprevisibilidade em uma dada situação concreta de trabalho e em um determinado contexto cultural".

No caso específico da área de ES, essas competências e habilidades estão relacionadas com a capacidade que o estudante deve desenvolver para realizar atividades de desenvolvimento de software (SAVI, 2011) (NUNES, 2015). Nesse contexto, o currículo de referência da ACM/IEEE (2013) sugere que os graduandos da área de Computação devem obter algumas competências técnicas em ES, estando essas competências relacionadas aos tópicos das unidades de conhecimento.

Dessa forma, as competências técnicas consideradas nesse trabalho são retiradas do Modelo de Competências em Engenharia de Software (*Software Engineering Competency Model* – SWECOM) (IEEE COMPUTER SOCIETY, 2014A). Esse modelo se baseia nas diretrizes curriculares de cursos de graduação em Engenharia de Software (ACM/IEEE, 2014) e no Guia para o Corpo de Conhecimento em Engenharia de Software (*Guide to the Software Engineering Body of Knowledge* – SWEBOK) (IEEE COMPUTER SOCIETY, 2014B).

O enfoque principal desse trabalho são as competências técnicas, organizadas em áreas que, por sua vez, são detalhadas em habilidades. Cada habilidade é descrita em termos de atividades. As atividades são especificadas em 5 (cinco) níveis de competência (IEEE COMPUTER SOCIETY, 2014A):

1. Técnico;
2. Profissional de Nível Iniciante;
3. Profissional;
4. Líder Técnico;
5. Engenheiro de Software Sênior.

O nível de competência esperado para os alunos que seguirem o modelo proposto é o Profissional de Nível Iniciante, que consiste em ter habilidades para auxiliar na execução de uma atividade ou executar uma atividade com supervisão. De acordo com o SWECOM (IEEE COMPUTER SOCIETY, 2014A), um indivíduo que possui as competências de um Profissional de Nível Iniciante deve possuir o conhecimento equivalente ao que é fornecido por um curso de graduação ou o equivalente a quatro ou cinco anos de experiência relevante na área.

2.4.3 Estilos de Aprendizagem

De acordo com Gimenes (2015, p. 22), a Engenharia de Software enquanto disciplina da Computação deve apropriar-se de teorias e práticas pedagógicas contemporâneas e conscientizar-se do perfil de seus estudantes, para planejar as práticas de ensino-aprendizagem. Conhecer o estilo de aprendizagem de uma pessoa permite ao professor orientar o aluno de acordo com o seu método preferido. Quanto mais assertivo for o estímulo, maior será o aprendizado, uma vez que a base de conhecimento é adquirida conforme as principais habilidades de cada aluno.

De acordo com França et al. (2016), os instrumentos mais comuns para identificar os estilos de aprendizagem são: a Teoria de Aprendizado de Kolb (1984) e o Modelo VARK (FLEMING e MILLS, 1992).

A. Teoria de Kolb

A teoria de aprendizagem experimental de Kolb (1984) define um ciclo de aprendizagem, conforme apresenta a **Figura 2**.

Figura 2 – Ciclo de Aprendizagem de Kolb (1984)

Fonte: Adaptado de Gary et al. (2013).

Segundo Kolb (1984), para aprender efetivamente, deve-se planejar e praticar o que se pretende aprender (Experimentação Ativa). Em seguida, deve-se fazer algo, ou seja, ter uma experiência (Experiência Concreta). Após essa experiência concreta, o aluno deve revisar e refletir sobre o que foi feito (Observação Reflexiva). Por fim, deve-se realizar uma conclusão sobre o aprendizado obtido a partir da experiência (Conceptualização Abstrata). Contudo, a aprendizagem efetiva somente ocorre quando o aprendiz está habilitado a executar esses 4 (quatro) estágios.

Quanto aos estilos de aprendizagem, Kolb (1984) pontua que pessoas diferentes naturalmente preferem certos estilos de aprendizagem diferentes. Contudo, o estilo de aprendizagem de cada indivíduo pode ser representado através da combinação dos seus dois estilos preferidos. A representação dessa combinação se dá através de uma matriz 2 x 2 (dois por dois) dos estilos de aprendizagem, como ilustrado na **Figura 3**.

Figura 3 – Estilos de Aprendizagem de Kolb (1984)

	FAZER Experimentação Ativa	OBSERVAR Observação Reflexiva
SENTIR Experiência Concreta	Acomodação	Divergência
PENSAR Conceituação Abstrata	Convergência	Assimilação

Fonte: Adaptado de Kolb (1984).

Assim, os 4 (quatro) estilos de aprendizagem são (KOLB, 1984):

- **Divergência (sentir e observar):** pessoas hábeis em observar as coisas de diferentes perspectivas. Elas preferem observar a fazer, tendendo a obter informação e usar a imaginação. Desempenham melhor situações que requerem geração de ideias, por exemplo, *brainstorms*. Pessoas com esse estilo preferem trabalhar em grupos, ouvir com uma mente aberta e receber *feedback*;
- **Assimilação (observar e pensar):** pessoas que preferem uma explanação boa e clara ao invés de uma oportunidade prática. Elas são mais atraídas por teorias que pareçam lógicas do que por abordagens baseadas em valores práticos. Sobressaem-se em entender informações complexas e em organizá-las de forma clara e lógica. Pessoas com esse estilo são importantes para a eficiência no uso da informação e nas carreiras científicas. Em situações de aprendizagem formal, preferem leituras, palestras, explorar modelos analíticos e ter tempo para pensar soluções através dessas práticas;
- **Convergência (fazer e pensar):** pessoas que podem resolver problemas, usando a sua aprendizagem para encontrar soluções para questões práticas. Elas são mais atraídas para tarefas técnicas e problemas ao invés de questões interpessoais ou sociais. Pessoas com esse estilo possuem habilidades especialistas e técnicas. Gostam de experimentar através de novas ideias, simular e trabalhar com aplicações práticas;
- **Acomodação (fazer e sentir):** pessoas que agem por instinto interno ao invés de agir por análise lógica. Elas consideram a análise de outras pessoas e preferem uma abordagem prática e experimental. Pessoas com esse estilo são úteis em papéis que requerem ação e iniciativa. Elas preferem trabalhar em equipe para completar tarefas. Costumam fixar alvos e trabalhar ativamente, tentando diferentes caminhos para atingir um objetivo.

O ciclo iterativo adotado pelo modelo de ensino proposto nessa tese é fortemente baseado no ciclo de aprendizagem de Kolb (1984). Consequentemente, as abordagens de ensino selecionadas em cada uma das etapas do modelo contemplam os 4 (quatro) estilos de aprendizagem.

B. Modelo VARK

A fim de operacionalizar o conceito de estilos de aprendizagem, através da preferência individual por diferentes tipos de estímulos sensoriais, Fleming e Mills (1992)

propuseram o modelo VARK, acrônimo de *Visual*, *Auditory*, *Read/Writing* e *Kinesthetic* (em português: Visual, Auricular, Textual e Cinestésico). Cada um desses termos são categorias sensoriais que os seres humanos empregam no aprendizado e no processamento de informações.

De acordo com Fleming e Mills (1992), a maioria dos indivíduos apresenta múltiplas preferências, com estilos diferentes que se sobressaem em situações de aprendizagem diferentes. Esse estilo é denominado Multimodal, que consiste na habilidade de absorver as informações transmitidas de acordo com os seus diferentes perfis de preferência. Dessa forma, o indivíduo pode se adaptar melhor, por exemplo, a diferentes professores, materiais, métodos, técnicas e abordagens de ensino.

No intuito de identificar a preferência de estilos de aprendizagem, Fleming e Mills (1992) propuseram o Questionário VARK. A partir da aplicação desse questionário, França et al. (2016) realizaram um *survey* com 119 engenheiros de software para identificar seus estilos de aprendizagem. Os pesquisadores observaram que há uma tendência desses engenheiros de expressarem preferência de aprendizagem, respectivamente, através de estímulos: cinestésicos (corporais ou práticos), auditivos, visuais e textuais.

Observa-se que o modelo proposto incorpora abordagens de ensino que contemplam esses 4 (quatro) tipos de estímulos, por exemplo, através da realização de projetos práticos (cinestésicos), discussões (auditivos), uso de jogos e videoaulas (visuais) e leitura de artigos (textuais). Dessa forma, pretende-se atingir uma ampla variedade de estilos de aprendizagens, pois a aplicação de um estímulo predominante poderia beneficiar apenas um subconjunto dos alunos e prejudicar outro, em função dos diferentes estilos de aprendizagem deles.

É importante destacar que essa abordagem de estilos de aprendizagem não é totalmente aceita pelos pesquisadores da área de ensino. De acordo com Paul (2017), há uma grande diferença entre a maneira como alguém prefere aprender e o que realmente leva a uma aprendizagem eficaz e eficiente. Adicionalmente, quase todos os estudos que relatam evidências de estilos de aprendizagem não conseguem satisfazer os critérios fundamentais para a validade científica. Apesar dessa limitação, a adoção dessa abordagem no modelo não é determinante para a sua implementação, sendo adotado apenas para mapear as estratégias de ensino adotadas e os diferentes estilos descritos por Fleming e Mills (1992).

2.4.4 Níveis de Aprendizagem

A fim de detalhar os diferentes níveis de habilidades cognitivas, as diretrizes curriculares da ACM/IEEE para cursos de graduação em Engenharia de Software (ACM/IEEE, 2014) especificam uma terminologia baseada na taxonomia de Bloom (1956) que consiste em conhecimento, entendimento e aplicação.

Sendo assim, podem-se destacar 3 (três) níveis de aprendizagem:

- **Conhecer:** lembrar do material previamente ensinado. Esse nível diz respeito à capacidade de observação e recuperação da informação pelo aluno, isto é, se ele "traz à mente a informação apropriada";
- **Compreender:** entender a informação e o significado do material apresentado. Por exemplo, se o aluno é capaz de traduzir o conhecimento a um novo contexto, interpretar fatos, comparar, ordenar, agrupar, inferir causas e predizer consequências;
- **Aplicar:** usar o material aprendido em situações novas e concretas. Por exemplo, se o aluno usa informação, métodos, conceitos, teorias para resolver problemas que requerem as habilidades e conhecimento apresentados.

É importante destacar que, conforme especificam Nunes, Yamaguti e Nunes (2016), "Aplicar" engloba "Compreender" que, por sua vez, engloba "Conhecer". Deste modo, o modelo proposto considera o nível "Aplicar" que é aderente ao nível de competência "Profissional de Nível Iniciante" do SWECOM (IEEE COMPUTER SOCIETY, 2014A), conforme destacado na Subseção 2.4.2.

2.5 PRÁTICAS DE CAPACITAÇÃO DA INDÚSTRIA

Muitos profissionais, em sua maioria formados em cursos de Computação, não possuem conhecimento adequado em determinadas áreas de processo (como por exemplo, Gerência de Configuração) (SHAW, 2000). Dessa forma, as empresas acabam por realizar cursos, *workshops*, *mentoring* ou *coaching* para repasse técnico das práticas específicas que compreendem as áreas de processo a fim de nivelar o conhecimento das equipes.

Uma vez que a indústria se queixa de que os cursos de graduação não formam profissionais da maneira apropriada e adota suas próprias práticas de capacitação, o modelo proposto nessa tese busca antecipar a aplicação dessas, consideradas efetivas pela

indústria, ainda na formação desses profissionais. As práticas adaptadas e integradas com as etapas do modelo foram *mentoring* e *coaching*.

Essas duas práticas visam o aperfeiçoamento, sendo o *mentoring* mais voltado para a área profissional e o *coaching* um alinhamento da vida pessoal com a profissional. No *mentoring*, ao contrário do *coaching* em que são definidos metas e objetivos, não é estabelecido um prazo, pois a condução é feita conforme a evolução do mentorando. Adicionalmente, no *coaching*, o *coach* não diz ao seu cliente o que fazer, e não precisa ser necessariamente mais velho ou ter mais experiências que o mesmo, pois a sua função é a de despertar habilidades e não de transferir conhecimento.

2.5.1 *Mentoring*

A mentoria é um processo natural e informal que, ao longo do tempo, foi se transformando em algo mais estruturado. Cada mentor tem um estilo de mentoria próprio, mas é comum entre eles o uso de metáforas, analogias, histórias de personagens de sucesso, exemplos, demonstrações, dentre outras técnicas.

O processo de mentoria se desdobra em 4 (quatro) fases que se caracterizam pela evolução da relação entre mentor (responsável pela aplicação da prática) e mentorando (principal beneficiado pela prática) (GOMES et al., 2015):

1. **Fase de Iniciação:** acontece a partir do primeiro contato entre mentor e mentorando, quando ocorrem as apresentações. São estabelecidos os objetivos da mentoria, os papéis de cada um, uma explanação sobre o processo de mentoria, resultando em uma proposta de trabalho a ser realizada;
2. **Fase de Cultivo:** acontece o aconselhamento profissional. O mentor inicia ouvindo as dificuldades e objetivos profissionais do mentorando. A escuta permite traçar um perfil comportamental e profissional do mentorando. Assim, o mentor passa a ter uma percepção mais apurada e a empregar aconselhamentos a fim de encorajar e dar autonomia para que o mentorando possa tomar suas decisões profissionais;
3. **Fase de Separação:** objetiva a autonomia do mentorando, onde o mentor se ausenta para que o mentorando possa colocar em prática sua proposta de trabalho;

4. **Fase de Redefinição:** objetiva avaliar a evolução do mentorando, a partir da identificação de pontos fortes, pontos fracos e oportunidades de melhoria relacionadas à execução da proposta de trabalho.

O mentor deve possuir experiência profissional e de vida que, provavelmente, o mentorando deseje alcançar um dia. Por esse motivo, e por envolver aconselhamento e inspiração, sugere-se que profissionais da indústria possam orientar os alunos-alvo da aplicação do modelo, através de relatos de experiência, problemas recorrentes e apresentação de soluções técnicas (ver Subseção 4.4.3), de forma semelhante à prática de *mentoring*. Para a mentoria da unidade de Engenharia de Requisitos, por exemplo, pode-se solicitar a contribuição de um aluno egresso da instituição de ensino que atue na indústria como Analista de Requisitos.

Nesse sentido, o SWECOM (IEEE COMPUTER SOCIETY, 2014A) destaca que além das atividades técnicas, uma competência adicional que todos os engenheiros de software devem possuir é a capacidade de instruir e orientar outros na aplicação de métodos e técnicas e no uso de ferramentas para realizar essas atividades.

2.5.2 *Coaching*

O *coaching* tem se mostrado eficiente para melhoria de desempenho, de aprendizagem e aquisição de competências. Essa prática é baseada na conversação entre duas ou mais pessoas – *coach* (responsável pela aplicação da prática) e *coachee* (beneficiado pela prática) - e ocorre principalmente por meio de perguntas que levam as pessoas a refletirem, ajudando-as a liberar potencial, a aprender e, assim, a maximizar sua performance.

As etapas básicas do processo de *coaching* são (GOMES et al., 2015):

1. **Indagação:** deve-se iniciar a indagação por meio de perguntas significativas. As perguntas constituem a principal intervenção do *coach*. Para isso, ele deve fazer perguntas “a partir de uma atitude de curiosidade e de quem desconhece o que está acontecendo”;
2. **Definição de objetivos e metas:** visa compreender o estado atual do *coachee*, estabelecer o foco e então definir os objetivos e as metas que serão alvo do processo. Adicionalmente, definem-se as tarefas para atingir esses objetivos e metas;

3. **Preparação para mudanças:** busca estimular novas experiências e aprendizagens das quais o *coachee* possa tirar lições sobre como desenvolver ou atingir o que deseja. Nessa etapa o papel do *coach* é apoiar o *coachee* para que se sinta motivado com aquilo que deseja, criando uma perspectiva diferente, útil e ampliada;
4. **Plano de ação e metas:** diz respeito ao alcance do estado ou resultado desejado. Em geral, consiste na elaboração de um plano de ação e na definição de metas exequíveis. A intenção é levar o *coachee* a assumir um compromisso com a ação e com a mudança que pretende realizar;
5. **Apoio à transição:** o *coachee* deve ter tempo para absorver o novo aprendizado ou desenvolver novas competências. Durante esse processo, ele deve fornecer *feedback* constante ao *coach*.

Segundo Gomes et al. (2015), é possível que o papel do *coach* seja realizado por um gestor, que atue como facilitador da aprendizagem, devendo para tal: conquistar a confiança do *coachee*; inspirar o *coachee* em prol de propósitos; definir objetivos estimulantes; estabelecer metas desafiantes; fornecer *feedback*; apoiar nas atividades.

Esse perfil se assemelha bastante ao desempenhado por professores que adotam abordagens humanistas de ensino (ver Subseção 2.3.2). Portanto, sugere-se que os professores realizem um papel semelhante ao do *coach* durante o ensino das unidades de conhecimento da ES. Adicionalmente, a prática de *coaching* define o indivíduo como foco, acredita na capacidade de mudança e incentiva a ação, principalmente a partir de indagações pertinentes. Sendo assim, o professor poderá exercer *coaching* principalmente durante a etapa de Iniciação do modelo iterativo de ensino (ver Subseção 4.4.1).

2.6 CONCLUSÃO

Esse capítulo apresentou a fundamentação teórica da tese através da exposição das principais definições e conceitos aqui abordados. A partir da revisão da literatura, optou-se por adotar a definição de modelo de ensino de Joyce e Weil (1972). Após a análise dos principais elementos do modelo proposto, observou-se que o mesmo se classifica como uma abordagem humanista, principalmente devido ao seu foco no aluno.

Esse trabalho também optou por adotar a classificação da ACM/IEEE (2013), que define a Engenharia de Software como uma área de conhecimento organizada em um conjunto de unidades de conhecimento que, por sua vez, são constituídas por tópicos e

aprendizagens esperadas. Sendo o principal objetivo desse trabalho o desenvolvimento de competências técnicas na área de ES, apresentaram-se diversas definições para esse termo e quais conceitos relacionados foram adotados como referências para essa pesquisa.

Adicionalmente, discutiu-se sobre os diferentes estilos de aprendizagem dos alunos, que impactam no desenvolvimento dessas competências, e os níveis que essa aprendizagem pode alcançar. Por fim, abordaram-se duas práticas atuais de capacitação adotadas pela indústria de software: co*aching* e *mentoring*. Essas práticas permitem tanto a aquisição de novas competências quanto o desenvolvimento das já existentes.

3 MÉTODOS DE PESQUISA

3.1 INTRODUÇÃO

Um dos principais objetivos desta pesquisa é encontrar evidências de que abordagens focadas no aluno e práticas de capacitação da indústria são mais adequadas para o desenvolvimento de competências técnicas. Para tal, definiu-se um modelo que incorpora essas abordagens e práticas a fim de apoiar o ensino de ES, a partir de um ciclo iterativo que contempla diferentes perfis e níveis de aprendizagem. Esse modelo foi avaliado a fim de observar se o ensino de ES baseado em abordagens práticas desenvolve competências técnicas em nível de aplicação, de maneira mais adequada, do que abordagens tradicionais de ensino centradas no professor e em aulas didáticas/expositivas.

Buscando atender a esses objetivos, este capítulo apresenta o *framework* metodológico da pesquisa (Seção 3.2), que trata de uma análise das questões de pesquisa e do posicionamento filosófico do pesquisador. Apresenta, também, um detalhamento dos tipos de validade (Seção 3.3) que impactam a confiança empírica deste estudo. Em seguida, descreve o desenho de pesquisa (Seção 3.4), ou seja, os métodos empíricos utilizados e como os dados e resultados desses se relacionam. Cada método é descrito a partir do seu *design*, participantes, resultados, análise e discussões e ameaças à validade. Esse capítulo descreve os métodos que serviram para fundamentar a definição do modelo. Os métodos adotados para avaliar o modelo são descritos no Capítulo 5. Por fim, a Seção 3.7 apresenta as considerações finais desse capítulo, destacando a contribuição de cada método e como os resultados desses se complementam.

3.2 *FRAMEWORK* METODOLÓGICO

De acordo com Easterbrook et al. (2007), um dos primeiros passos para escolher um método de pesquisa apropriado é esclarecer as perguntas de pesquisas. Nos estágios iniciais de uma pesquisa, normalmente é necessário fazer perguntas exploratórias, a fim de tentar entender os fenômenos estudados, e identificar distinções úteis que esclareçam a compreensão do pesquisador.

Inicialmente, esta pesquisa faz perguntas descritivas e classificatórias (ver Seção 3.5), pois busca-se tanto identificar os tópicos contemplados pelos professores quanto

classificar as abordagens de ensino adotadas por estes em disciplinas da área de ES. Métodos de pesquisa adequados a perguntas exploratórias tendem a serem aqueles que oferecem dados qualitativos, já que ajudam a construir teorias experimentais (EASTERBROOK et al., 2007).

Posteriormente, realizaram-se perguntas relacionais (ver Seção 3.6) a fim de mapear os tópicos de currículos de referência e as práticas específicas de modelos de qualidade e assim relacionar o conteúdo de ES com as práticas adotadas pela indústria. A partir deste mapeamento, realizaram-se perguntas processo-descritivas a consultores da indústria a fim de levantar quais práticas são por eles adotadas durante a capacitação de profissionais da área de ES.

Após a especificação do modelo de ensino, realizaram-se perguntas do tipo *design* (ver Seção 5.2) a fim de avaliar a qualidade da documentação e usabilidade do modelo através de um painel de especialistas composto por 3 (três) doutores que atuam como professores/pesquisadores na área de ensino de ES.

Por fim, analisou-se qual tipo de abordagem de ensino prepara os alunos de forma mais adequada para o mercado. Dessa forma, foram feitas perguntas do tipo causalidade-comparação (ver Seção 5.3), pois busca-se saber se determinadas abordagens focadas no aluno desenvolvem, de forma mais adequada, competências técnicas do que as práticas tradicionais. Para esse tipo de perguntas, recomenda-se o uso de métodos quasi-experimentais (EASTERBROOK et al., 2007). Neste tipo de desenho de pesquisa, existem grupos de controle ou diversas medidas a serem observadas. Por esse motivo, não é possível fazer alocação aleatória dos sujeitos envolvidos. Devido a essas particularidades do fenômeno estudado, é importante destacar que o tipo de desenho escolhido impacta diretamente na validade interna ou na capacidade de se estabelecer relação causal.

Ainda segundo Easterbrook et al. (2007), após especificar uma pergunta de pesquisa, vale a pena considerar o que se aceita como uma resposta válida, pois diferentes pessoas fazem diferentes suposições sobre a verdade científica. Essa postura filosófica em relação à verdade científica afeta quais métodos se acredita que levem à evidência aceitável em resposta à sua pergunta de pesquisa (CRESWELL, 2002).

Sendo assim, o posicionamento filosófico adotado nessa pesquisa será positivista que, segundo Creswell (2002), defende que todo conhecimento deve ser baseado em

inferência lógica a partir de um conjunto básico de fatos observáveis. Os positivistas acreditam que o conhecimento científico é construído de forma incremental a partir de observações verificáveis, e inferências com base nessas observações. Além disso, os positivistas acreditam que métodos mistos podem ser utilizados a fim de reforçar as conclusões de suas pesquisas (HESSE-BIBER, 2010).

3.3 TIPOS DE VALIDADE

De acordo com Travassos, Gurov e Amaral (2002), a questão fundamental a respeito dos resultados dos métodos de pesquisa é quão válidos são eles. Assim, os resultados devem ser válidos para a população da qual o conjunto de participantes foi recebido. É importante também para generalizar os resultados para uma população mais ampla. Neste sentido, os resultados possuem a validade adequada se são válidos para a população a qual tendem ser generalizados.

Existem 4 (quatro) tipos de validade para os resultados de métodos de pesquisa: validade de conclusão, validade interna, validade de construção e validade externa.

A validade de construção considera os relacionamentos entre a teoria e a observação, ou seja, se o tratamento reflete bem a causa e se o resultado reflete bem o efeito (TRAVASSOS, GUROV e AMARAL, 2002). Já a validade interna define se o relacionamento observado entre o tratamento e o resultado é causal, ou seja, não é resultado da influência de outro fator (não controlado ou medido) (TRAVASSOS, GUROV e AMARAL, 2002). Assim, no que diz respeito à validade interna deste trabalho, as ameaças estão relacionadas à definição do desenho de pesquisa (ver Seção 3.4) e à seleção das amostras e das medidas adotadas para representar as variáveis sob observação.

A validade de conclusão está relacionada à habilidade de chegar a uma conclusão correta a respeito dos relacionamentos entre o tratamento e o resultado do experimento (TRAVASSOS, GUROV e AMARAL, 2002). Nesse sentido, consiste na probabilidade de a hipótese nula ter sido rejeitada corretamente. Essa rejeição está intimamente associada ao tamanho de amostra. Sendo assim, nesta pesquisa, as ameaças estão relacionadas à escolha do tamanho da amostra e à análise dos dados estatísticos. Por fim, a validade externa define condições que limitam a habilidade de generalizar os resultados de um experimento para o contexto da indústria (TRAVASSOS, GUROV e AMARAL, 2002).

3.4 DESENHO DE PESQUISA

Conforme destacado na metodologia de pesquisa (Seção 1.4), o desenho de pesquisa desta tese é baseado em uma abordagem multi-métodos (HESSE-BIBER, 2010), combinando um *survey*, um painel de especialistas e um experimento controlado. Nesse desenho de pesquisa, é usada a técnica de triangulação[1] para consolidar os resultados obtidos nos diferentes métodos, considerando que questões de pesquisa relacionadas serão investigadas nesses métodos. Dessa forma, a triangulação reforça as conclusões e completude do estudo, trazendo maior credibilidade aos resultados da pesquisa.

Baseado no posicionamento positivista, optou-se por selecionar o método *Survey* para responder às perguntas descritivas e classificatórias. De acordo com Kitchenham e Pfleeger (2008), *survey* é um método empírico abrangente para a coleta de informações que visa descrever, comparar ou explicar conhecimentos e comportamentos de uma população. Dessa forma, entende-se que a finalidade de um *survey* é produzir estatísticas, ou seja, descrições quantitativas ou numéricas de alguns aspectos da população de estudo.

Easterbrook et al. (2007) destaca que o método empírico *survey* identifica as características de uma ampla população de indivíduos a partir de um subconjunto representativo dessa população. Nesse sentido, os resultados de um *survey* recaem quase que exclusivamente na tradição positivista, pois o desejo de caracterizar uma população inteira através de técnicas de amostragem requer certa crença no reducionismo e preocupação com teorias generalizáveis (CRESWELL, 2002).

A fim de responder as perguntas relacionais e as processo-descritivas, adotaram-se dois processos de pesquisa não empíricos: um mapeamento e um levantamento. O mapeamento foi realizado através da análise da estrutura de componentes dos currículos de referência com a estrutura dos modelos de qualidade. Já o levantamento foi realizado através da aplicação de um questionário estruturado, que foi respondido por 10 (dez) consultores que realizam a implantação de MPS.

As perguntas do tipo *design* foram respondidas através de um painel de especialistas (BEECHAM et al., 2005). Esse método empírico é utilizado ou quando não

[1] A triangulação combina métodos de coleta de dados qualitativos e quantitativos, assim como diferentes métodos de análise dos dados. Seu objetivo é reforçar os resultados dos dados analisados, considerando que todo método de pesquisa possui suas fraquezas.

existe unanimidade de opinião sobre determinado assunto, ou para resolver as discordâncias, possibilitando o desenvolvimento de consenso.

Para responder às perguntas do tipo causalidade-comparação desta pesquisa, optou-se por selecionar o método Experimento Controlado (WOHLIN et al., 2012). Um experimento controlado é uma investigação de uma hipótese testável onde uma ou mais variáveis independentes são manipuladas para medir sua efetividade em relação a uma ou mais variáveis dependentes (EASTERBROOK et al., 2007). Esse método permite determinar como as variáveis se relacionam e se existe relação de causa-efeito entre elas.

O experimento controlado está intimamente relacionado à postura positivista, uma vez que é essencialmente reducionista (EASTERBROOK et al., 2007). Nesse método, reduz-se a complexidade através da seleção de algumas variáveis de interesse que serão modificadas, de uma forma controlada, enquanto controla-se todas as outras variáveis (WOHLIN et al., 2012).

3.5 SURVEY

3.5.1 *Design*

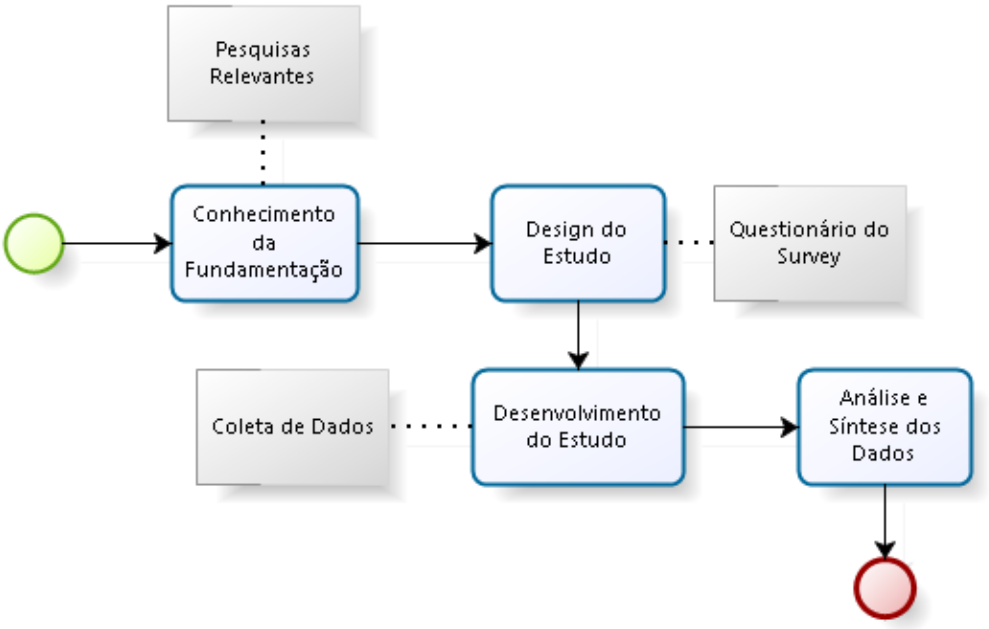
O *survey* realizado objetivou coletar informações sobre as opiniões de alunos e professores, representados por uma amostra dessas populações, em relação ao ensino de ES. Esse *survey* foi aplicado em cursos de graduação em Computação de universidades públicas e privadas do Brasil e seguiu os *guidelines* de Kitchenham e Pfleeger (2008). A análise dos resultados desse *survey* encontra-se no Apêndice A desta tese.

Esse *survey* se classifica, quanto ao *design* de coleta de dados, como *cross-sectional*, onde a coleta dos dados ocorre em um momento único. Nesse tipo de *survey*, os participantes são questionados sobre suas experiências passadas em um ponto fixo no tempo (KITCHENHAM e PFLEEGER, 2008).

A fim de planejar e executar o *survey* desta pesquisa, foram seguidas 4 (quatro) etapas, conforme ilustra a **Figura 4**. Inicialmente, na *Etapa I – Conhecimento da Fundamentação* foram realizadas pesquisas sobre a relevância de tópicos de ES, especificamente os trabalhos de Wangenheim e Silva (2009) e Lethbridge (2000) foram estudados em profundidade como forma de consolidar a base conceitual necessária para as próximas etapas do *survey*. Na *Etapa II – Design do Estudo* realizou-se o planejamento do *survey* a fim de identificar a população-alvo, as questões de pesquisas e os métodos de

análise de dados. Nessa etapa, identificaram-se os tópicos de ES sugeridos pelos currículos de referência da ACM/IEEE (2013) e as estratégias adotadas para o ensino de ES (PRIKLADNICKI et al., 2009).

Figura 4 – Etapas da Realização do *Survey*



Fonte: Adaptado de Kitchenham e Pfleeger (2008).

Realizou-se na *Etapa III – Desenvolvimento do Estudo* realizou-se a coleta de dados. Durante o período de 16 de Março a 29 de Maio de 2015, questionários foram divulgados para cerca de 50 professores e mais de 350 alunos, em listas de e-mails, grupos da área de ES em redes sociais, e *in loco* em universidades públicas e privadas. Por fim, na *Etapa IV – Análise dos Dados e Síntese* buscou-se identificar quais tópicos de ES eram considerados mais relevantes para o ensino de ES, de acordo com a adoção destes pelos professores e o grau de interesse dos alunos, quais abordagens de ensino os professores adotavam e quais dessas os alunos consideravam mais efetivas para o seu aprendizado.

Foram utilizados como instrumentos para aplicação do *survey* 2 (dois) questionários auto administrados, com questões de pesquisa que tinham um conjunto de respostas pré-definidas. Essas questões de pesquisa foram fortemente baseadas nos *surveys* aplicados por Wangenheim e Silva (2009) e Lethbridge (2000). Revisando os currículos de referência da ACM/IEEE (2013) e da SBC (2005), identificaram-se 83 tópicos de ES e 125 aprendizagens esperadas, classificados e organizados em 10 (dez) unidades de conhecimento (ver Tabela 3).

Antes da coleta de dados, foi realizado um pré-teste de aplicação do *survey* com 2 professores e 4 alunos de instituições públicas e privadas a fim de identificar possíveis inconsistências e o tempo necessário para responder a todas as questões de pesquisa dos questionários. Os participantes desse pré-teste indicaram que o tempo para responder todas as questões foi, em média, 32 minutos.

Os questionários foram disponibilizados na *web*, utilizando a ferramenta *Google Forms* (*https://www.google.com/forms*), que permite a coleta de informações através de uma pesquisa personalizada que é automaticamente ligada a uma planilha. Os questionários encontram-se disponíveis em *http://goo.gl/vn5jHS*. A divulgação desses questionários se deu através de listas de e-mails, grupos da área de ES em redes sociais, e *in loco* em universidades públicas e privadas.

A. Survey com Professores da Disciplina de Engenharia de Software

A **Tabela 4** mostra as 3 (três) questões descritivas e classificatórias feitas para os professores da disciplina de Engenharia de Software a fim de identificar quais tópicos e quais abordagens de ensino estão sendo adotados.

Tabela 4 – Questões e Respostas do *Survey* com Professores

Questão	Opções de Respostas
Q1. *Quais currículos de referências são adotados na definição da ementa da disciplina de Engenharia de Software?*	( ) *Curriculum Guidelines* da ACM/IEEE; ( ) Currículo de Referência da SBC ; ( ) Outros; ( ) Nenhum.
Q2. *Quais tópicos de Engenharia de Software estão sendo contemplados na ementa destas disciplinas?*	Para cada um dos 83 tópicos de ES, o professor deveria responder: ( ) Contemplado ( ) Não Contemplado
Q3. *Quais abordagens de ensino são adotadas na disciplina de Engenharia de Software?*	A. Métodos de Ensino; B. Abordagens de Ensino; C. Estratégias de Avaliação.

Fonte: Portela, Vasconcelos e Oliveira (2015D).

B. Survey com Alunos Concluintes da Disciplina de Engenharia de Software

A **Tabela 5** mostra as 3 (três) perguntas descritivas e classificatórias feitas para os alunos concluintes da disciplina de Engenharia de Software a fim de analisar a aprendizagem e sua preferência por abordagens de ensino. Para as questões Q1 e Q2 foi

utilizada a escala *Likert* de 0 até 5 para caracterizar os graus de utilidade e aprendizagem, respectivamente.

Tabela 5 – Questões e Respostas do *Survey* com Alunos

Questão	**Opções de Respostas**
Q1. *O quão útil considera a disciplina de Engenharia de Software para a sua formação profissional?*	0. Completamente inútil 1. Quase nunca útil 2. Ocasionalmente útil 3. Moderadamente útil 4. Muito útil 5. Essencial
Q2. *O quanto aprendeu das unidades de conhecimento ministradas durante a disciplina de Engenharia de Software?*	Para cada uma das 125 aprendizagens de ES, o aluno deveria responder: 0. Não aprendi absolutamente nada 1. Aprendi vagamente 2. Aprendi o básico 3. Aprendi moderadamente 4. Aprendi muito 5. Aprendi em profundidade
Q3. *Quais abordagens de ensino de Engenharia de Software considera melhor para sua aprendizagem?*	A. Abordagens de Ensino; B. Estratégias de Avaliação.

Fonte: Portela, Vasconcelos e Oliveira (2015D).

3.5.2 Participantes

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

I. Professore(a)s que lecionam/lecionaram disciplinas da área de ES a partir de 2005 (ano de publicação do currículo de referência da SBC);

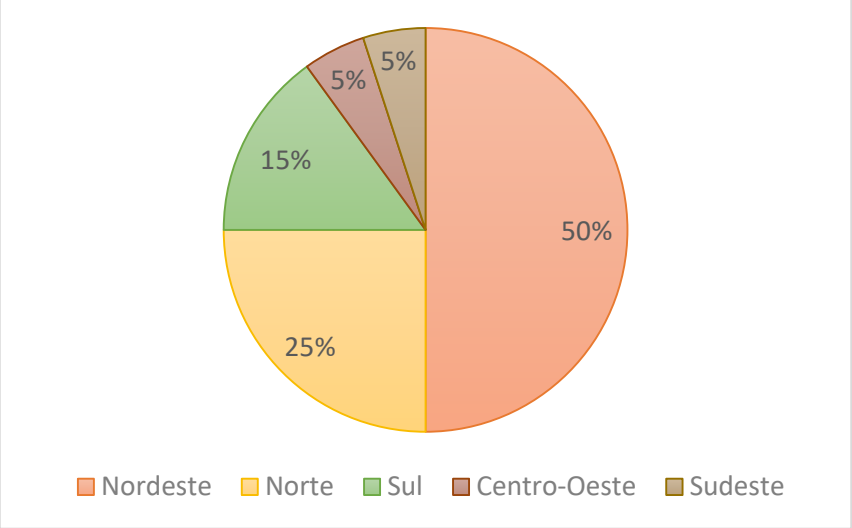
II. Aluno(a)s que concluíram disciplinas da área de ES entre 2005 e 2015 (período de vigência do currículo de referência da SBC e de realização do estudo).

Foram excluídos os participantes que 1) não atendiam a esses critérios, 2) não estavam motivados a participar da pesquisa ou, ainda, 3) aqueles que claramente não conseguiram responder as questões de pesquisa. Esses critérios de inclusão e exclusão foram divulgados no questionário do *survey*.

O *survey* foi divulgado para mais de 50 professores e mais de 350 alunos. No entanto, muitos destes não responderam (segundo relatos) devido ao tempo estimado para

análise e preenchimento. Apenas 23 professores e 47 alunos responderam todas as perguntas. Assim, receberam-se respostas de 70 participantes de uma ampla variedade de instituições. Apesar da baixa amostra, os participantes representam 12 estados do Brasil, cerca de 44% do total de estados. Destes, 50% são de instituições da região Nordeste, 25% do Norte, 15% do Sul, 5% do Centro-Oeste e 5% do Sudeste, conforme sintetiza o **Gráfico 1**.

Gráfico 1 – Percentual de Instituições Participantes por Região



Fonte: Elaborado pelo autor.

Obtiveram-se respostas de 20 instituições de ensino públicas e privadas, sendo a maioria dos respondentes, 80%, de instituições públicas de ensino e 20% de instituições privadas. A instituição que teve o maior número de participantes foi a Universidade Federal do Pará (UFPA), da qual participaram 3 professores e 20 estudantes. Quanto à região com maior participação, destaca-se o Nordeste, do qual participaram 10 instituições de ensino.

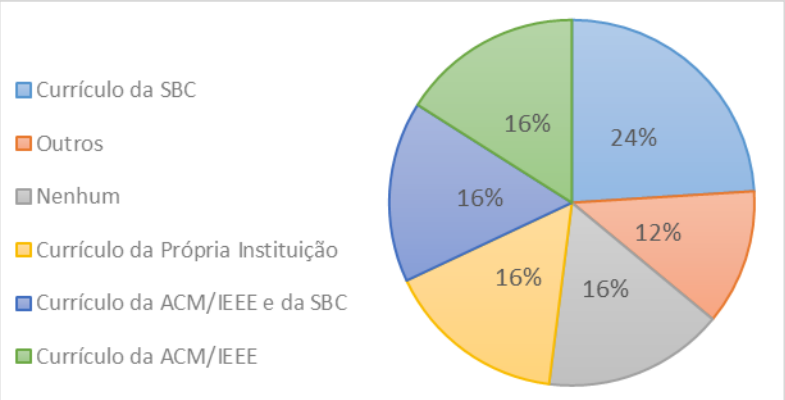
A média de idade dos professores é de 38 anos. Em média, o ano da última formação destes professores foi 2010, sendo a formação mais recente em 2015 e a mais antiga em 1998, há 17 anos. Quanto à formação, 4% possui Especialização e a maioria, 39%, possuem Mestrado ou Doutorado. 19% dos professores entrevistados possuem pós-doutorado. Em média, estes professores lecionam a disciplina de Engenharia de Software há 8 anos, sendo o maior tempo de atuação 25 anos e o menor 1 ano.

A média de idade dos alunos é de 24 anos. Em média, o ano de conclusão da disciplina foi 2013, sendo a formação mais recente em 2015 e a mais antiga em 2005, há 10 anos.

3.5.3 Resultados

Em relação à Questão 1, aos currículos de referência adotados pelos professores, observa-se que a maioria (24%) adota o currículo de referência da SBC. Há um equilíbrio quanto os demais, pois 16% utiliza o currículo da ACM/IEEE e outros 16% adota esses 2 currículos. 16% dos professores adota um currículo definido pela própria instituição e 12% considera outras abordagens, como SWEBOK e ENADE. Por fim, 16% dos professores não adota nenhum currículo de referência na definição de suas ementas. Esses resultados são sintetizados no **Gráfico 2**.

Gráfico 2 – Currículos de Referência adotados pelos Professores

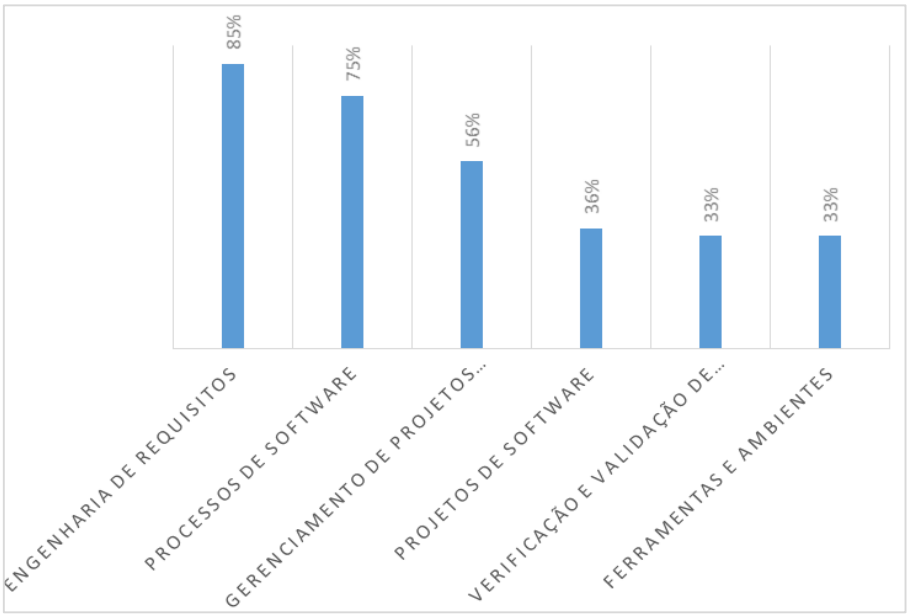


Fonte: Elaborado pelo autor.

Quanto à Questão 2, os tópicos contemplados pelos professores nas ementas das disciplinas de ES, é possível analisar o percentual de tópicos adotados para cada unidade de conhecimento. No **Gráfico 3**, observa-se que a Engenharia de Requisitos possui a maior relevância, ou seja, 85% de seus tópicos são adotados pelos professores entrevistados.

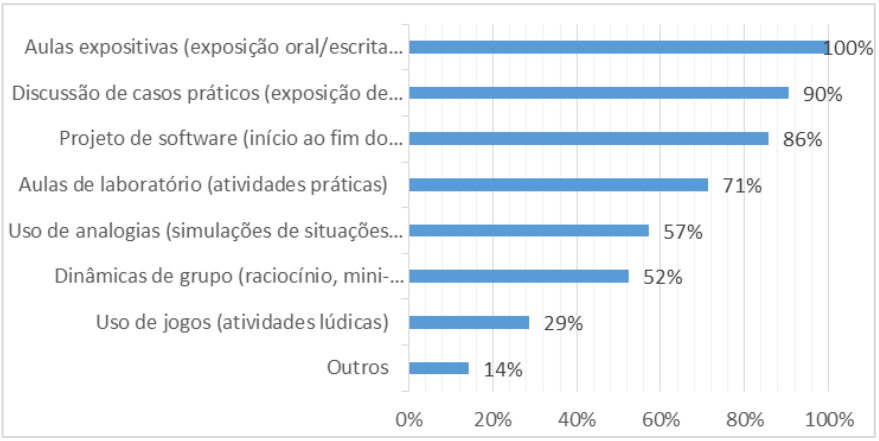
Em relação à Questão 3, as abordagens de ensino adotadas pelos professores, observou-se que todos adotam aulas expositivas (100%) e a maioria discute casos práticos com os alunos (90%), realizam projetos de software (86%) e ministram aulas de laboratório (71%), conforme apresenta o **Gráfico 4**.

Gráfico 3 – Percentual de Tópicos adotados por Unidade de Conhecimento



Fonte: Elaborado pelo autor.

Gráfico 4 – Abordagens de Ensino adotadas pelos Professores



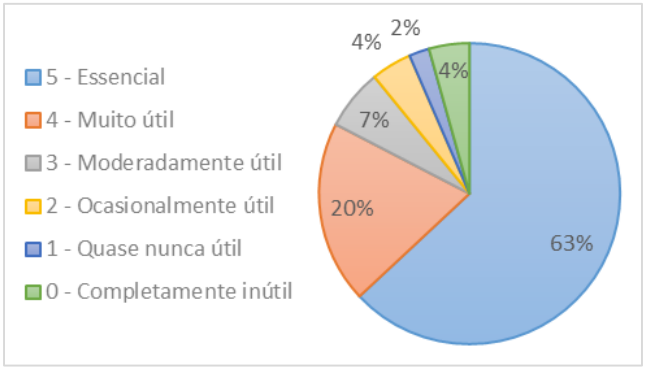
Fonte: Elaborado pelo autor.

Quanto aos resultados do *survey* com os alunos, para a Questão 1, relacionada à percepção da utilidade da área de ES para a formação profissional deles, 63% considera a disciplina essencial, 20% muito útil e 7% moderadamente útil, conforme **Gráfico 5**. Observa-se que os demais 10% dos alunos entrevistados não considera a área muito útil, sendo 4% ocasionalmente, 2% quase nunca e 4% acham que a ES é completamente inútil.

Para a Questão 2, relacionada à aprendizagem dos tópicos das unidades de conhecimento, é possível analisar a média percentual de aprendizagem dos alunos para cada unidade de conhecimento. No **Gráfico 6**, observa-se que a Engenharia de Requisitos

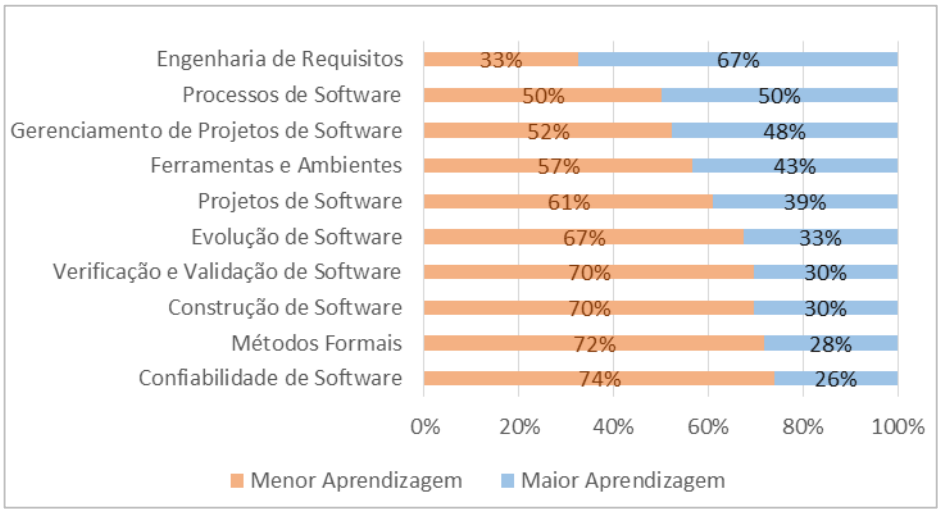
possui a maior porcentagem de aprendizagem, ou seja, 67% de seus tópicos são mais aprendidos (igual ou maior a 3 pontos na escala *Likert*) pelos alunos.

Gráfico 5 – Percepção da Utilidade da Área de Engenharia de Software



Fonte: Elaborado pelo autor.

Gráfico 6 – Média Percentual de Aprendizagem por Unidade de Conhecimento

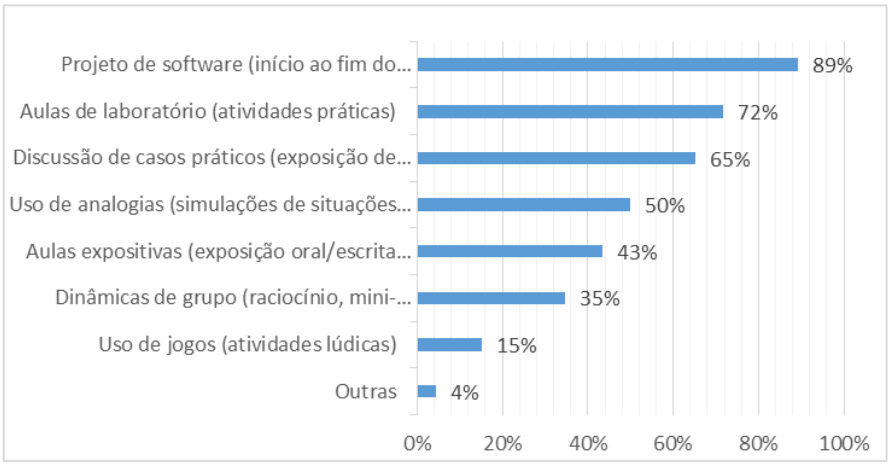


Fonte: Elaborado pelo autor.

Além da Engenharia de Requisitos, as unidades Processos de Software, Gerenciamento de Projetos de Software e Ferramentas e Ambientes possuem grande aprendizagem, acima de 40%. Já as unidades Verificação e Validação de Software, Construção de Software, Métodos Formais e Confiabilidade de Software apresentam menor aprendizagem (abaixo de 3 pontos na escala *Likert*), abaixo de 30%.

Por fim, quanto à Questão 3, relacionada à percepção da efetividade das abordagens de ensino adotadas pelos professores, observa-se que a grande maioria dos alunos considera a realização de Projetos de software, Aulas de laboratório e Discussão de casos práticos. O percentual destas e de outras abordagens é apresentado no **Gráfico 7**.

Gráfico 7 – Percepção de Efetividade das Abordagens de Ensino



Fonte: Elaborado pelo autor.

3.5.4 Análise e Discussões

A. Sobre a Adoção de Currículos de Referência

É de suma importância a adoção destes currículos de referência, pois são discutidos e elaborados por órgãos representativos da área como a Sociedade Brasileira da Computação (SBC) e ACM/IEEE. Além disso, definem uma estrutura de conceitos inter-relacionados a fim de especificar diretrizes de ensino e formação profissional de acordo com as áreas da Computação, sendo uma delas a Engenharia de Software. Estas diretrizes definem o perfil profissional e acadêmico esperado para os estudantes da área, bem como estruturação e detalhamento de matérias, como carga horária, tópicos a serem abordados e aprendizagens esperadas para cada um destes tópicos.

Sem a adoção destes currículos, os professores acabam por estabelecer ementas incompatíveis com diretrizes nacionais e internacionais de ensino e, possivelmente, não atendendo o que se espera de um curso de graduação em computação. Como não há diretrizes curriculares, não é possível comparar a qualidade da ementa destes professores com o restante daqueles que adotam estes currículos da SBC e ACM/IEEE.

B. Sobre a Utilidade e Aprendizagem da Engenharia de Software

De acordo com a ACM/IEEE, a Engenharia de Software se constitui como uma das áreas de grande relevância nos cursos da área de Computação. Isto decorre tanto da importância do processo de software, objeto de estudo desta disciplina, em si quanto dos desafios relacionados com a formação completa de um profissional que irá atuar no mercado de software. Grande parcela destes alunos, após a formação, irá atuar em um

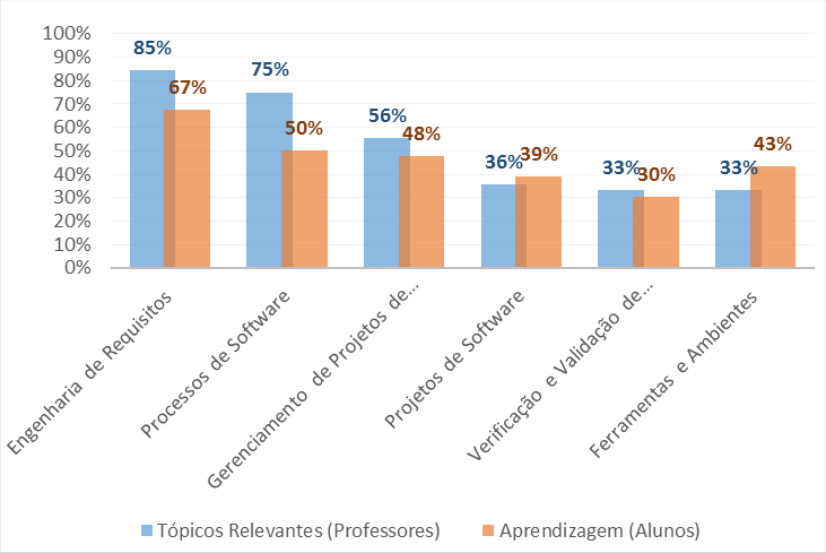
contexto de processo de desenvolvimento de software, sendo necessário conhecimento relacionado aos conceitos e tópicos relacionados a esta área.

No que diz respeito à aprendizagem, os alunos apresentaram um baixo percentual para a resposta “Aprendi em profundidade” (apenas 2%) e uma grande porcentagem para “Não aprendi absolutamente nada” (15% das respostas dos alunos). Isto acaba por reforçar as constatações da indústria de que os alunos não estão tendo uma formação adequada. Em particular, na Engenharia de Software, a grande quantidade de tópicos acaba por favorecer este cenário, onde os alunos não conseguem profundidade no aprendizado de determinados tópicos.

C. Sobre a Relevância das Unidades de Conhecimento

De 10 unidades de conhecimento da Engenharia de Software, destacam-se as 6 que possuem tópicos mais relevantes, ou seja, aqueles que são mais adotados pelos professores nas ementas das disciplinas. A partir da identificação destas unidades, é possível correlacionar o percentual de tópicos relevantes com o percentual de aprendizagem dos alunos, conforme o **Gráfico 8**.

Gráfico 8 – Correlação entre Tópicos Mais Relevantes e Aprendizagem



Fonte: Elaborado pelo autor.

Observa-se que a unidade Engenharia de Requisitos, de acordo com os resultados deste *survey*, é que possui maior relevância, pois é amplamente contemplada nos currículos de Engenharia de Software, por 85% dos professores, e efetivamente aprendida por 67% dos alunos entrevistados. Em seguida, destacam-se as unidades de Processos de Software, ministrada por 75% dos professores e aprendida por 50% dos alunos, e

Gerenciamento de Projetos de Software, ministrada por 56% dos professores e aprendida por 48% dos alunos entrevistados. Para as demais unidades relevantes, ouve um certo desalinhamento entre relevância e aprendizagem. Por exemplo, 36% dos professores abordam Projetos de Software, porém 39% dos alunos aprendem efetivamente esta unidade, um percentual menor do que a aprendizagem da unidade Ferramentas e Ambientes, 43%, que é abordada apenas por 33% dos professores.

Acredita-se que a relevância da unidade Engenharia de Requisitos deve-se ao fato de que a Engenharia de Software é uma disciplina preocupada com o desenvolvimento efetivo e eficiente de sistemas de software que satisfaçam os requisitos dos usuários. Para satisfazer as necessidades dos usuários, é de extrema importância o conhecimento de técnicas de elicitação, modelagem e escrita de requisitos, tópicos estes abordados nesta unidade.

Além disto, os profissionais 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. Neste contexto, insere-se a segunda unidade mais relevante, Processos de Software, o principal objeto de estudo da Engenharia de Software. De acordo com as aprendizagens esperadas para esta unidade, os alunos devem ser capazes de definir, modelar e executar um processo de software.

No entanto, não basta definir um processo e um plano de projeto, é necessário pôr em prática em fazer o acompanhamento deste para realizar ajustes caso necessário. Neste contexto, se insere a terceira unidade mais relevante, Gerenciamento de Projetos de Software. O Projeto de Software consiste em instanciar um Processo a fim de atender um plano específico, que detalha um escopo, cronograma, equipe e riscos. A relevância desta unidade deve-se a importância e dificuldade de realizar a gerência de um Projeto e, principalmente, da equipe que irá executá-lo. Observa-se, assim, que a gerência se mostra mais importante que o próprio projeto, segundo os professores e alunos entrevistados.

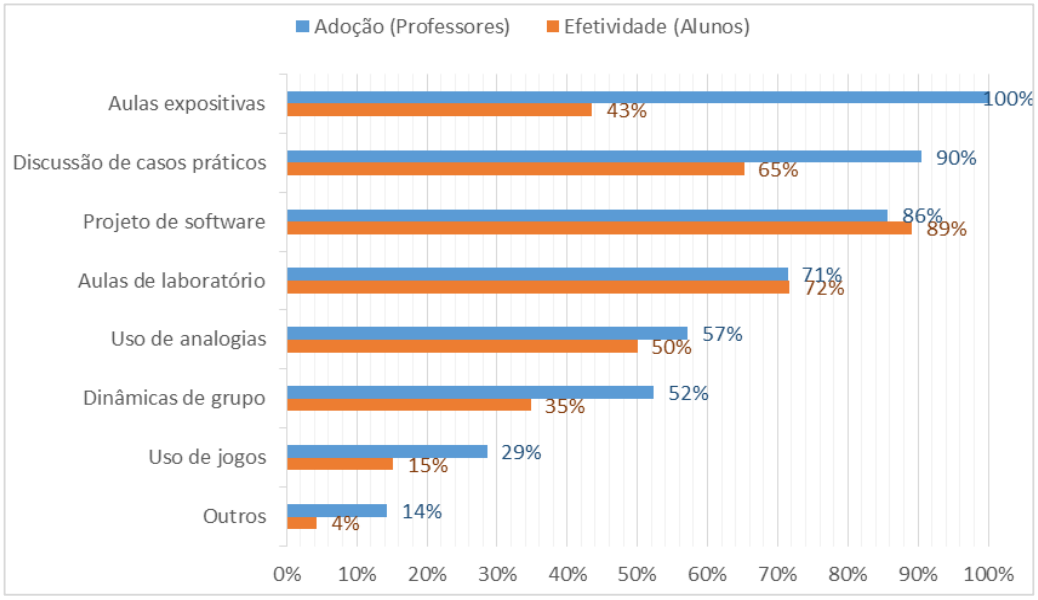
Por fim, quanto as três unidades restantes, observa-se uma complementariedade. A unidade de Projetos de Software consiste em adotar padrões de projetos, arquiteturas, interfaces, qualidade a fim de atender um conjunto específico de Requisitos seguindo um Processo de Software. Como dito anteriormente, o Gerenciamento de Projetos é a unidade responsável pela sua execução e acompanhamento. A unidade de Verificação e Validação complementa a unidade de Projeto na medida em que é responsável pelos testes e auditorias de qualidade do processo executado e dos produtos de trabalho gerados. Por

fim, a unidade Ferramentas e Ambientes apoia a definição, modelagem e execução de Processos, bem como a execução de Projetos na medida que permite realizar testes e controle de versões, por exemplo.

D. Sobre a Adoção e Efetividade das Abordagens de Ensino

Quanto as abordagens de ensino, estabelece-se uma correlação entre a adoção destas pelos professores e o quanto os alunos consideram estas efetivas para seu aprendizado. O **Gráfico 9** apresenta o resultado desta correlação.

Gráfico 9 – Correlação entre Abordagens de Ensino Adotadas e Efetividade



Fonte: Elaborado pelo autor.

Apesar de todos professores realizarem Aulas expositivas, apenas 43% dos alunos consideram estas efetivas para seu aprendizado. É uma porcentagem menor do que a atribuída para Uso de analogias, adotada por menos de 70% dos professores. Também houve uma certa discordância em relação à Discussão de casos práticos, adotado por 90% dos professores e considerado efetivo por 65% dos alunos entrevistados. Houve convergência nas abordagens de Projeto de software e Aulas de laboratório.

A partir dos resultados deste *survey*, observa-se que os alunos estão interessados em realizar atividades práticas, como projetos de desenvolvimento em laboratórios que simulem situações próximas as que vão encontrar no mercado. Acredita-se que isso se deve ao fato da disciplina Engenharia de Software possuir muitos tópicos, 83 no total, o que acaba por torná-la menos atrativa para alunos que não possuem uma afinidade com a

área. A abordagem prática permite aos alunos fixar melhor os conceitos a partir da sua aplicação.

3.5.5 Ameaças à Validade

Existem diversos vieses a serem considerados na realização deste *survey*. Inicialmente, destaca-se que as amostras obtidas, conforme destacado na Subseção 3.5.2, foram baixas, o que afeta a validade de conclusão. O *survey* foi divulgado para cerca de 50 professores e mais de 350 alunos. Entretanto, apenas 70 responderam, sendo 23 professores e 47 alunos. Isso corresponde a uma taxa de 46% de resposta dos professores e apenas 13% dos alunos. Adicionalmente, destaca-se que essa baixa amostra da população do *survey* tende a reduzir a validade externa, pois os resultados obtidos sobre esses alunos e professores podem não se aplicar a outras instituições de ensino.

A fim de contornar essa baixa amostragem e, consequentemente, tratar a validade de conclusão e externa, optou-se por selecionar participantes que representassem uma parcela representativa da população-alvo. Nesse sentido, os participantes do *survey* representam 12 estados do Brasil, sendo 50% desses de instituições da região Nordeste, 25% do Norte, 15% do Sul, 5% do Centro-Oeste e 5% do Sudeste. A maioria dos participantes, 80%, é oriunda de instituições públicas de ensino e 20% de instituições privadas.

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. 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 aprendizagem dos alunos com os questionários dos *surveys* conduzidos por Lethbridge (2000) e Wangenheim e Silva (2009).

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 e Pfleeger (2008). Assim, pode-se garantir que as etapas necessárias foram seguidas na definição do *design* e na condução da pesquisa.

3.6 MAPEAMENTO E LEVANTAMENTO

3.6.1 *Design*

A partir do contexto desta pesquisa, no qual a indústria está insatisfeita com os profissionais recém-formados (LETHBRIDGE et al., 2007), identificou-se a necessidade de levantar quais estratégias as empresas utilizam para capacitar suas equipes. Dadas as problemáticas expostas na Seção 1.2, principalmente no que diz respeito às limitações no ambiente acadêmico para a realização de projetos práticos de desenvolvimento, optou-se por realizar um mapeamento entre os tópicos dos currículos de referência da ACM/IEEE (2013) e SBC (2005) e as práticas específicas dos modelos de qualidade *Capability Maturity Model Integration for Development* (CMMI-DEV) (SEI, 2010) e Modelo de Referência MPS para Software (MR-MPS-SW) (SOFTEX, 2012).

Esse mapeamento tem por objetivo relacionar os tópicos de ES com as boas práticas adotadas pela indústria no processo de desenvolvimento de software. Modelos de qualidade como o CMMI-DEV e o MR-MPS-SW fundamentam-se em normas da área e apresentam um conjunto de áreas de processo/processos e práticas específicas/resultados esperados que servem de referência para empresas de software.

A fim de realizar esse mapeamento, utilizou-se como base o currículo da ACM/IEEE (2013) que, conforme relatado na Subseção 2.4.1, abrange os Tópicos da SBC (2005). De maneira semelhante, optou-se por usar como base o CMMI-DEV (SEI, 2010), pois esse serviu de referência e possui aderência com o MR-MPS-SW (SOFTEX, 2012). Assim, realizou-se a seguinte pergunta relacional: *"Qual é a relação entre os tópicos de ES do currículo da ACM/IEEE e as práticas específicas do modelo CMMI-DEV?"*. Os dados foram coletados durante o período de 21 de Julho à 3 de Setembro de 2015.

O detalhamento completo desse mapeamento encontra-se no Apêndice B desse trabalho. Por fim, destaca-se que foram mapeadas apenas as unidades de conhecimento identificadas como de maior relevância na etapa do *Survey* (PORTELA, VASCONCELOS e OLIVEIRA, 2015D).

No que diz respeito à capacitação, destacam-se as estratégias adotadas por consultores em MPS que possuem experiência no desenvolvimento de competências e habilidades profissionais em equipes que serão submetidas à avaliação para certificação em modelos de qualidade (ROCHA, 2014). Nesse sentido, consideram ser possível

adequar determinadas práticas da indústria para o contexto de uma disciplina da área de ES. No entanto, há escassez de professores qualificados na academia, que tenham de fato atuado na indústria (GARG e VARMA, 2008).

A fim de atender a essa demanda, bem como identificar práticas da indústria e estratégias de capacitação, realizou-se um levantamento com consultores em Melhoria do Processo de Software (MPS). O objetivo desse levantamento, disponível em *https://goo.gl/FSfsiQ*, foi identificar, junto a especialistas da área (consultores de MPS e implementadores de modelos de qualidade que também atuem como professores de graduação), quais as estratégias de capacitação e avaliação são adotadas em programas de MPS. A **Tabela 6** mostra as perguntas processo-descritas feitas aos consultores nesse levantamento.

Tabela 6 – Perguntas do Levantamento de Práticas de Capacitação da Indústria

Pergunta	**Respostas Identificadas**
Quais modelos de melhoria do processo de software você implementa?	( ) MR-MPS-SW ( ) CMMI-DEV ( ) Outros
Qual a estratégia de capacitação adotada para os processos relacionados à Engenharia de Software?	Para cada unidade de conhecimento da ES: ( ) *Workshop* ( ) Dinâmica ( ) *Mentoring* ( ) *Coaching* ( ) Outros
Como o aprendizado é avaliado nestas estratégias?	( ) Prova Objetiva ( ) Prova Subjetiva ( ) Participação ( ) Frequência ( ) Outros

Fonte: Elaborado pelo autor.

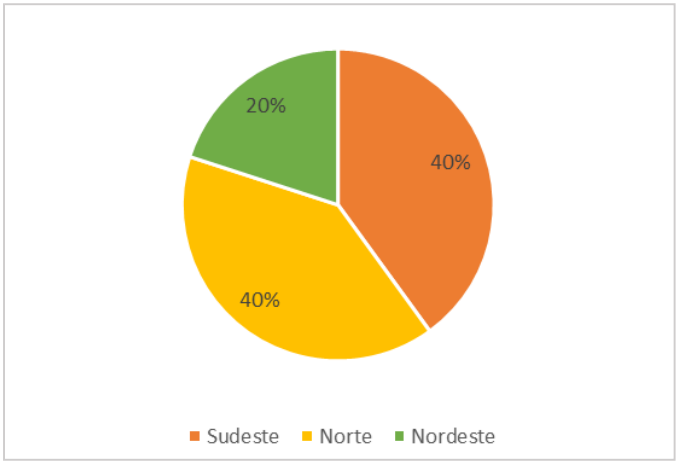
3.6.2 Participantes

O mapeamento foi realizada pelos autores deste trabalho, 3 (três) consultores em MPS que possuem experiência tanto na implementação do modelo CMMI-DEV quanto com o currículo da ACM/IEEE. A média de atuação profissional em consultoria dos participantes desse mapeamento é de 9 anos.

Em relação ao levantamento, receberam-se respostas completas de 10 consultores. É uma baixa amostra, considerando que o mesmo foi enviado para cerca de 40 consultores. A média de tempo que estes consultores atuam como professores é de 13 anos e meio, sendo o menor tempo de docência 6 meses e o maior 32 anos. Já a média de tempo que estes consultores atuam como implementadores de MPS é de 11 anos, sendo o menor tempo de consultoria 4 anos e o maior 32 anos.

Quanto a região das instituições onde esses consultores atuam como professores, Norte e Sudeste tiveram 40% de participantes cada uma e 20% na região Nordeste, conforme apresenta o **Gráfico 10**.

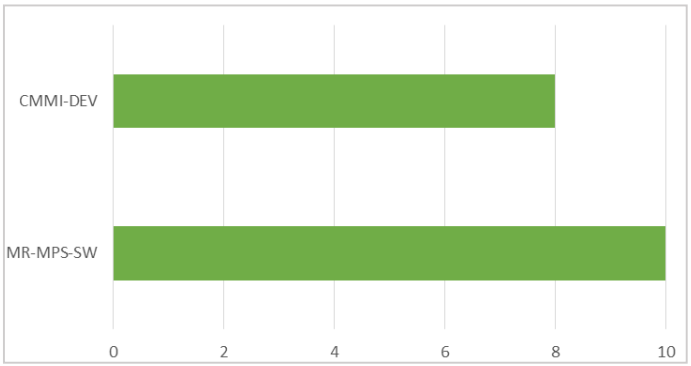
Gráfico 10 – Regiões das Instituições de Ensino dos Professores/Consultores



Fonte: Elaborado pelo autor.

Quanto aos modelos de MPS, 100% dos consultores já implementaram o Modelo de Referência MPS para Software (MR-MPS-SW) e 80% já implementaram o *Capability Maturity Model Integration for Development* (CMMI-DEV), conforme **Gráfico 11**.

Gráfico 11 – Modelos de MPS adotados por Professores/Consultores

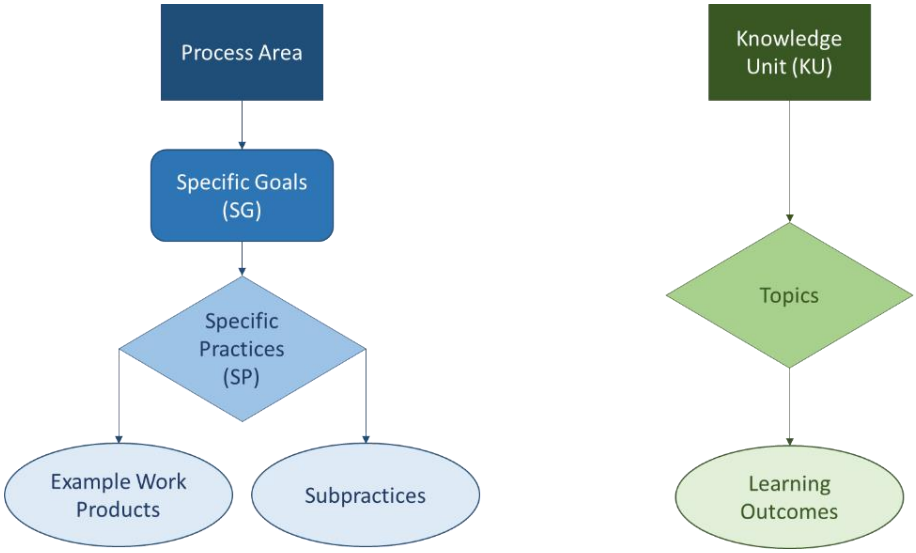


Fonte: Elaborado pelo autor.

3.6.3 Resultados

A primeira etapa do mapeamento ocorreu em nível de estrutura e conceitos, conforme apresentam, respectivamente, a **Figura 5** e a **Tabela 7**.

Figura 5 – Mapeamento entre o Modelo CMMI-DEV e o Currículo da ACM/IEEE



Fonte: Elaborado pelo autor.

Tabela 7 – Mapeamento entre o Modelo CMMI-DEV e o Currículo da ACM/IEEE

Conceito do CMMI-DEV	Conceito da ACM/IEEE	Descrição do Mapeamento
Área de Processo (*Process Area)*	Unidade de Conhecimento (*Knowledge Unit*)	Uma área de processo é um conjunto de práticas relacionadas e uma unidade de conhecimento é um conjunto de tópicos relacionados.
Objetivo Específico (*Specific Goals*)	<<Não identificado>>	Não há um objetivo definido para um conjunto de tópicos.
Prática Específica (*Specific Practices*)	Tópicos (*Topics*)	Uma prática específica é a descrição de uma atividade e um tópico é a descrição de um conhecimento.
Subpráticas (*Subpractices*)	Aprendizagem Esperada (*Learning Outcomes*)	Uma subprática é uma descrição detalhada de uma prática específica e uma aprendizagem esperada está relacionada a um tópico específico.

Fonte: Elaborado pelo autor.

Uma área de processo é um conjunto de práticas relacionadas que, quando implementadas coletivamente, satisfazem a um conjunto de objetivos considerados importantes para melhoria de uma área, como, por exemplo, Gerência de Requisitos

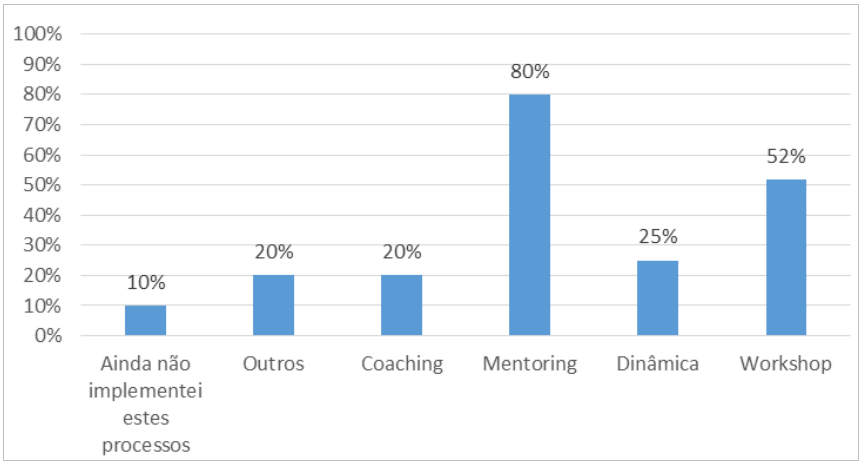
(*Requirements Management* - REQM). De maneira similar, uma unidade de conhecimento é um conjunto de tópicos relacionados que compõem o conhecimento de uma área da ES, como por exemplo, Engenharia de Requisitos (*Requirements Engineering*).

Uma prática específica é uma descrição de uma atividade que é considerada importante para o atendimento de um objetivo específico associado, como por exemplo "SP 1.1 Levantar Necessidades". Já um tópico é a descrição de um conhecimento relacionado a uma unidade, como por exemplo, "Elicitação de requisitos de software".

Por fim, uma subprática é uma descrição detalhada que providencia orientação para interpretação e implementação de uma prática específica, como por exemplo "Estabelecer critérios objetivos para avaliação e aceitação de requisitos". Uma aprendizagem esperada está relacionada ao resultado do ensino de um tópico específico, como por exemplo "Realizar uma avaliação de um conjunto de requisitos de software para determinar a qualidade dos requisitos no que diz respeito às boas características de requisitos".

Em relação ao levantamento, a partir da análise das respostas dos consultores, identificaram-se as práticas mais adotadas, conforme mostra o **Gráfico 12**.

Gráfico 12 – Percentual de Adoção de Práticas de Capacitação



Fonte: Elaborado pelo autor.

No que diz respeito ao levantamento, a partir da análise das respostas obtidas, observou-se que a prática de capacitação mais adotada foi *mentoring* (80% dos entrevistados), que consiste no aconselhamento oriundo de indivíduos que detêm maior conhecimento e experiência que outros (GOMES et al., 2015). A segunda prática mais adotada foi *workshop* (52% dos entrevistados), onde os consultores realizam um

seminário de curta duração, apresentando técnicas e métodos e demonstrando como esses podem ser aplicados.

O uso de dinâmicas (25% dos entrevistados), onde um instrutor sugere uma atividade lúdica análoga a uma atividade técnica, e a prática de *coaching* (20% dos entrevistados), onde um profissional estimula e inspira outro a maximizar seu potencial, por meio do desenvolvimento de novos e mais efetivos comportamentos, ainda são poucos explorados por consultores. Destaca-se a necessidade da adoção dessas práticas, devido aos benefícios inerentes à sua implementação.

A partir da identificação e da análise dessas práticas, listadas no Apêndice C deste trabalho, foram discutidas as estratégias de adequação dessas para o ensino de ES no modelo de ensino proposto.

3.6.4 Análise e Discussões

A prática de capacitação mais adotada foi *mentoring*, onde os consultores orientam e compartilham com os profissionais da empresa-alvo da melhoria, suas experiências e conhecimentos nos processos de MPS. Essa prática foi adaptada para o papel do professor, que atuou como um mentor, mantendo e acompanhando o progresso educacional em sala de aula. Ao assumir essa função, é necessário que o professor realize questionamentos pertinentes e apresente aos alunos possíveis soluções. Desse modo, os alunos escolhem quais soluções serão adotadas, a fim de que possam aprender de acordo com essas escolhas.

A segunda prática mais adotada foi *workshop*, onde os consultores apresentam técnicas, habilidades e ferramentas, demonstrando como essas podem ser aplicadas. Essa prática foi adaptada através do convite a profissionais que atuam em determinadas áreas de conhecimento da ES para que esses possam contribuir com um relato de experiência sobre dificuldades e possíveis soluções técnicas relacionadas a essa área.

Observou-se que o uso de dinâmicas, apesar de pouco adotadas na indústria, possui um grande potencial de envolver os alunos e de explorar os tópicos específicos de ES. Para essa prática, uma dificuldade foi identificar sugestões de dinâmicas para cada unidade de conhecimento. De forma semelhante, a prática de *coaching*, apesar de adotada apenas por 20% dos consultores entrevistados, foi um ponto positivo relatado pelos alunos e pelos professores. Essa prática foi desempenhada por alunos veteranos

(monitores da disciplina) que realizaram treinamentos em determinada técnica ou ferramenta de apoio.

De acordo com Prikladnicki et al. (2009), os participantes de dinâmicas relatam que a interação entre os processos é melhor compreendida do que em exposições teóricas. Tais dinâmicas permitem aumentar o nível de interação ente os alunos e o facilitador, como também auxiliam na consolidação de conceitos a partir da vivência da teoria. Já a prática de *coaching* proporciona a um estudante veterano realizar um treinamento específico diretamente com o aluno. Nesse sentido, *coaching* talvez seja a prática mais efetiva do ponto de vista técnico.

3.6.5 Ameaças à Validade

O principal viés relacionado com o mapeamento e levantamento diz respeito à baixa amostra da população envolvida: 3 consultores realizaram a análise e comparação entre o currículo da ACM/IEEE e o modelo CMMI-DEV e 10 consultores responderam quais práticas adotam na capacitação de profissionais. Essa baixa amostragem pode afetar a validade de conclusão e validade externa.

Desta forma, a fim de reduzir este viés da amostragem através da seleção de uma amostra significativa da população de consultores, realizou-se o levantamento nas regiões norte, nordeste e sudeste do Brasil. No total, obtiveram-se respostas de 5 instituições implementadoras de MPS. A instituição implementadora de MPS que teve o maior número de participantes foi a SWQuality (Recife-PE), da qual participaram 5 consultores. Essa instituição foi responsável pela aplicação do modelo CMMI-DEV em 17 instituições no Brasil[2] (dados de Outubro de 2017).

3.7 CONCLUSÃO

Este capítulo apresentou os principais métodos de pesquisas e etapas deste trabalho. Inicialmente, apresentaram-se as classificações das perguntas de pesquisa, o posicionamento filosófico dos pesquisadores e os métodos empíricos mais adequados para responder a essas perguntas. Em seguida, apresentou os 4 (quatro) tipos de validade que devem ser verificados para os de métodos de pesquisa: validade de conclusão, validade interna, validade de construção e validade externa.

[2] https://sas.cmmiinstitute.com/pars/

Posteriormente, descreveu-se o desenho de pesquisa, composto por um *survey*, um mapeamento, um levantamento, um painel de especialistas e um experimento controlado. A partir da realização do *survey*, identificaram-se as unidades de conhecimento mais adotadas em disciplinas de ES, bem como os tópicos de cada uma dessas unidades. Essas unidades serão o foco inicial da instanciação do modelo. Além disso, identificaram-se as estratégias de ensino e avaliação mais adotadas pelos professores. Observou-se que essas estratégias, de maneira geral, não são consideradas efetivas pelos alunos.

O mapeamento e o levantamento foram realizados a fim de identificar as práticas adotadas por consultores que permitem o desenvolvimento de competências técnicas em unidades específicas da ES. Assim, identificou-se a relação entre essas unidades de conhecimento dos currículos de referência da área e os processos de modelos de qualidade adotados pela indústria. Posteriormente, consultores em Melhoria do Processo de Software informaram quais práticas de capacitação adotavam para implementação desses processos. Destacaram-se, dentre essas, as práticas de *coaching* e *mentoring*.

Já o painel de especialistas e o experimento controlado, descritos no Capítulo 5, foram executados a fim de avaliar o modelo apresentado no Capítulo 4.

4 MODELO ITERATIVO PARA O ENSINO DE ES

4.1 INTRODUÇÃO

Existem várias abordagens, técnicas, métodos e estratégias para o ensino de ES disponíveis na literatura. Portanto, não é o objetivo deste trabalho apresentar uma nova abordagem de ensino. Baseado em 7 (sete) princípios (Seção 4.2), o principal objetivo do modelo proposto é apoiar a aplicação de abordagens humanistas (apresentadas na Seção 4.3) de forma conjunta e iterativa no processo de ensino-aprendizagem de disciplinas relacionadas à área de ES. Também não se caracteriza como objetivo desse modelo atender especificamente a um curso de graduação em Engenharia de Software, mas sim o ensino de ES em qualquer disciplina de qualquer curso de graduação da área de Computação.

Assim, espera-se que os professores possam usufruir dos benefícios da aplicação integrada dessas abordagens. Adicionalmente, espera-se que os alunos possam se motivar e engajar mais a partir da cobertura das etapas do modelo por uma ampla variedade de estilos de aprendizagens. Como consequência do foco nos alunos, do uso de abordagens práticas e de práticas de capacitação da indústria, espera-se que os alunos participem ativamente do processo de ensino-aprendizagem e desenvolvam de forma mais eficiente competências técnicas na área de Engenharia de Software.

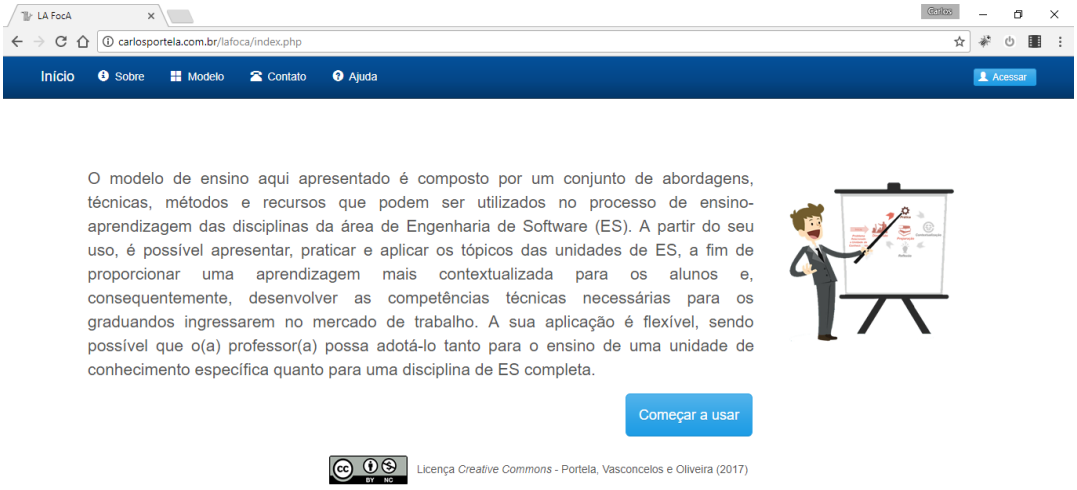
O modelo, conforme definição apresentada na Seção 2.2, fornece uma sintaxe (Seção 4.4) que visa orientar professores na seleção e organização de técnicas, estratégias e métodos de ensino. De maneira complementar, busca auxiliar a interação professor-aluno durante o processo de ensino-aprendizagem através de um sistema social (Seção 4.5). Nesse contexto, especificaram-se os requisitos e materiais de apoio necessários à aplicação do modelo em sala de aula (Seção 4.6).

Esse modelo é composto por uma especificação, um exemplo de uso (ver Apêndice G) e um sistema *web*[3]. A especificação consiste na documentação técnica que apresenta seus objetivos, sintaxe, fundamentação, aplicação e efeitos esperados. O exemplo de uso mostra como aplicar o modelo no ensino da unidade de Engenharia de Requisitos (ER). Já o sistema *web*, apresentado na **Figura 6**, busca sistematizar a

[3] http://carlosportela.com.br/lafoca/

aplicação do modelo em sala de aula, permitindo seguir o fluxo iterativo do modelo, acessar exemplos de materiais de apoio e registrar os dados dessa instanciação do modelo.

Figura 6 – Sistema *Web* do Modelo Iterativo de Ensino



Fonte: Elaborado pelo autor.

Tanto a especificação quanto o exemplo de uso estão disponíveis no sistema, todos sob licença *Creative Commons*[4], sendo permitido o seu uso não comercial com a devida atribuição dos direitos aos autores deste trabalho.

4.2 PRINCÍPIOS PARA O ENSINO DE ES

A partir da análise das abordagens práticas, levantadas na revisão da literatura e destacadas na Seção 4.3 desse capítulo, os autores deste trabalho identificaram 7 (sete) princípios que norteiam o ensino prático de Engenharia de Software:

I. **O aluno é o foco do processo de aprendizagem:** o ensino deve ser focado no aluno, onde o professor deve atuar como um facilitador, guiando os aprendizes na busca de soluções. Dessa forma, os alunos tendem a adquirir conhecimento através da aprendizagem ativa e das relações com outros alunos;

II. **Os alunos têm estilos de aprendizagem diferentes:** cada aluno processa as informações de forma diferente, apresentando um estilo de aprendizagem predominante (cinestésico, auditivo, visual ou textual). Fornecer abordagens de ensino favoráveis a esses estilos tende a facilitar o processo de ensino-aprendizagem dos alunos;

[4] https://br.creativecommons.org/

III. **A aprendizagem deve ser baseada na resolução de problemas:** o uso de problemas baseados no mundo real estimula os alunos a desenvolver o pensamento crítico, criarem habilidades para solução de problemas reais e a adquirir conhecimento sobre os principais conceitos estudados;

IV. **A realização de projetos práticos permite aplicar conhecimento:** o envolvimento dos alunos em projetos práticos de desenvolvimento de software permite que esses possam aplicar o conhecimento obtido de maneira satisfatória, ou seja, desenvolver competências técnicas;

V. **A abordagem de ensino deve ser iterativa:** os alunos tendem a aprender os tópicos de ES de forma mais eficaz através de abordagens iterativas, pois terão a oportunidade de realizar suas atividades em um ciclo, avaliar o seu trabalho ao final deste e, em seguida, aplicar o conhecimento adquirido em um próximo ciclo;

VI. **A capacitação na indústria foca no desenvolvimento de competências:** profissionais que realizam capacitação na indústria de software tendem a adotar práticas que valorizam o desenvolvimento de competências técnicas. A aplicação dessas práticas no contexto acadêmico poderia potencializar a aquisição de novas competências pelos alunos e desenvolver as já existentes;

VII. **As competências devem ser refinadas em níveis:** devem-se identificar quais competências pretende-se que o aluno desenvolva e em qual nível. Geralmente, trabalha-se com 3 (três) níveis: conhecer (lembrar do material previamente ensinado); compreender (interpretar, comparar, entender o material); e aplicar (usar o material aprendido em situações práticas). O nível esperado para um aluno egresso que pretende atuar na área de ES é o de aplicação.

Esses princípios serviram de base para identificar os pontos comuns entre as diversas abordagens práticas descritas na literatura. A partir da análise desses pontos, pode-se realizar a integração dessas abordagens através de um ciclo iterativo, baseado na teoria de Kolb (1984), que organiza o fluxo de etapas do modelo proposto nesta tese. As principais abordagens práticas para o ensino de ES que seguem esses princípios são descritas na seção a seguir.

4.3 ABORDAGENS PRÁTICAS PARA O ENSINO DE ES

Os currículos de referência da ACM/IEEE (2013) e da SBC (2005) ressaltam que as competências emergem através do estudo teórico das unidades de ES e da aplicação prática de seus conceitos. Nesse sentido, é fundamental que os estudantes entendam a relação entre a teoria e prática e como uma influencia a outra. Porém, de acordo com Prikladnicki et al. (2009), as abordagens mais comuns para ensinar ES incluem aulas expositivas, aulas de laboratório, entre outras.

Diversos estudos já observaram que o ensino de ES predominantemente teórico tem mostrado resultados insuficientes em relação à compreensão dos conceitos e métodos estudados, para preparar adequadamente o profissional para atuar no mercado de trabalho, se esse não tiver previamente alguma experiência prática, por mais simples que seja (MONSALVE, WERNECK e LEITE, 2010) (PRIKLADNICKI et al., 2009) (GNATZ et al., 2003). No entanto, não existe atualmente uma abordagem predominante para conduzir essas experiências práticas em ES (MALIK e ZAFAR, 2012).

Inicialmente, a abordagem baseada em projetos práticos era amplamente adotada. Contudo, durante os últimos dez anos, outras abordagens alternativas vêm sendo propostas e adotadas para o ensino de ES (MARQUES, QUISPE e OCHOA, 2014). As subseções a seguir apresentam essas abordagens alternativas que realizam atividades práticas no ensino de ES, identificadas nos trabalhos relacionados a essa pesquisa e descritas a partir dos seus enfoques.

4.3.1 Realização de Projetos Práticos

A ACM/IEEE (2013) recomenda que os cursos de graduação em Computação envolvam os estudantes em projetos práticos de desenvolvimento de software para que eles possam aplicar o conhecimento obtido de maneira satisfatória. Assim, o estudante poderá aplicar seu conhecimento teórico, pois um projeto prático de desenvolvimento de software irá requerer que esse resolva problemas técnicos, trabalhe em equipe e desenvolva habilidades de comunicação interpessoal (SBC, 2005). Além disso, o desenvolvimento de software requer muitas habilidades, como definição do projeto, gerenciamento, programação, validação, análise, estudo dos usuários, documentação e técnicas específicas. Portanto, essas competências do engenheiro de software devem ser trabalhadas durante a graduação (GNATZ et al., 2003).

De maneira geral, os projetos práticos de cursos de graduação em Computação são aplicados a fim de preparar os estudantes para a prática profissional (GNATZ et al., 2003) (ACM/IEEE, 2013). Nesses projetos, tradicionalmente, os estudantes trabalham em grupos para desenvolver um software em reposta a um problema do mundo real. Gnatz et al. (2003) destacam que o principal objetivo dessa abordagem é providenciar aos estudantes a experiência de um projeto de desenvolvimento similar ao da indústria. Eles acreditam que a única maneira que realmente prepara alguém para fazer um trabalho efetivo na indústria é aprender fazendo.

Além disso, Ohlsson e Johansson (1995) defendem que a realização de um projeto prático de desenvolvimento permite que o graduando aprenda a gerenciar o tempo, prioridade e progressos. Outros benefícios da realização desses projetos incluem o trabalho em grupo, a avaliação de potenciais soluções, o desenvolvimento de habilidades de comunicação e de relações interpessoais (SBC, 2005) (ACM/IEEE, 2013).

Por fim, quanto à realização de projetos práticos, se faz necessário viabilizar, em ambientes acadêmicos, a execução de projetos de qualidade, semelhantes aos realizados na indústria, de modo a agregar valor aos alunos (CHEN e CHONG, 2011) (RODRIGUES e ESTRELA, 2012). Bessa, Cunha e Furtado (2012) destacam que o ensino da ES precisa de ferramentas que simulem ambientes da indústria e seus respectivos clientes a fim de minimizarem as lacunas de formação tais como: a identificação de papéis, o relacionamento com o cliente e os problemas relacionados a escopo, prazo, custo e qualidade.

Santos et al. (2009) adaptaram, em duas disciplinas de um curso de mestrado profissional, um ambiente semelhante a fábricas de software. Assim, os estudantes formaram equipes e ficaram imersos em um projeto prático de desenvolvimento de software, tentando resolver problemas de clientes reais, apoiados por processos, papéis e métricas para controlar os resultados alcançados. Como resultados, além do aumento nas notas, os autores afirmam que os alunos obtiveram grandes conhecimentos práticos e, consequentemente, ingressaram mais facilmente no mercado.

4.3.2 Aprendizagem Baseada em Problemas

Para Ohlsson e Johansson (1995), o objetivo de qualquer programa de educação em Engenharia de Software deve consistir em fornecer as habilidades para lidar com problemas da indústria. De maneira complementar, a ACM/IEEE (2013) ressalta que esse

objetivo pode ser atingido a partir da integração entre teoria e prática dentro de um currículo que promova a aquisição de conhecimento geral e específico para resolução de problemas reais da indústria de software.

Nesse sentido, existe um método utilizado na educação médica, desde a década de 70, denominado *Problem-Based Learning* (PBL) que, de maneira geral, pode ser definido como um método instrucional que usa um problema para iniciar, direcionar e motivar o aprendizado (BESSA, CUNHA e FURTADO, 2012). De maneira especifica, a PBL preza pelo uso de problemas baseados no mundo real para estimular os alunos a desenvolver pensamento crítico, habilidades para solução de problemas e adquirir conhecimento sobre os principais conceitos da área em questão (ANDRADE et al., 2010).

De acordo com Andrade et al. (2010), embora concebida para o ensino da medicina, a abordagem PBL vem sendo utilizada em outras áreas, como na Engenharia de Software. Essa aplicação na ES se dá por meio de um modelo adaptado, de forma parcial, dentro do currículo convencional, ou em partes de uma disciplina. Bessa, Cunha e Furtado (2012) ressaltam três importantes princípios do método PBL que podem auxiliar o ensino-aprendizagem de Engenharia de Software:

1. O aprendizado acontece em um ambiente onde os alunos estão imersos na prática, em atividades em que recebem *feedback* de seus colegas e professores;
2. Os alunos recebem guias e suporte de seus pares, de maneira a promover um ensino multidirecional envolvendo outros alunos e os professores, diferentemente do ensino convencional (professor para aluno);
3. O aprendizado é funcional, uma vez que parte de problemas reais.

É importante destacar que essa abordagem também é aderente ao Princípio I destacado na Seção 4.2, ou seja, a PBL é uma estratégia educacional focada no aluno, que o auxilia no desenvolvimento do raciocínio e comunicação, habilidades essenciais para o sucesso em sua vida profissional. Adicionalmente, nessa abordagem, o aluno é constantemente estimulado a aprender e a fazer parte do processo de construção do aprendizado (ANDRADE et al., 2010).

Santos, Alexandre e Rodrigues (2015) aplicaram PBL na disciplina de Gerenciamento de Projetos usando um método chamado xPBL desenvolvido pelo grupo de pesquisa NEXT (*iNnovative Educational eXperience in Technology*). O objetivo do xPBL é alinhar métodos e ferramentas ao gerenciamento da abordagem PBL para que os

princípios que o caracterizem possam ser garantidos quando adotados para o ensino de Computação. Assim, após a aplicação de xPBL nessa disciplina, solicitaram que os alunos avaliassem o grau de maturidade da PBL na disciplina segundo 10 (dez) princípios (SANTOS, FIGUERÊDO e WANDERLEY, 2013):

1. O(s) Problema(s) é(são) o núcleo da proposta educacional;
2. O aprendiz é o proprietário do problema;
3. A autenticidade do problema ou da tarefa é evidente;
4. A autenticidade do ambiente de aprendizagem é evidente;
5. A resolução do problema conduz o processo;
6. A complexidade do problema ou da tarefa reflete a realidade do mercado;
7. A avaliação e análise de como o problema foi resolvido seguem a metodologia;
8. A reflexão sobre o conteúdo aprendido e o processo de aprendizagem é feito regularmente e continuamente;
9. A aprendizagem colaborativa e multidirecional ocorre de forma contínua;
10. A avaliação contínua ocorre.

De acordo com a avaliação dos alunos, os princípios 3, 5, 6, 8, 9 e 10 foram bem desenvolvidos (média de 0.94), estando muito próximos do valor máximo (1.0). Quanto aos demais princípios, que receberam valores mais baixos (média de 0.80), Santos, Alexandre e Rodrigues (2015) destacam que esses se relacionavam aos perfis dos alunos e às limitações do ambiente acadêmico. Portanto, pouco poderiam ser explorados pelos professores e tutores.

4.3.3 Uso de Jogos e Simuladores

Entende-se por softwares educativos a concepção de ferramentas computacionais de suporte ao ensino, como jogos, simuladores e ambientes de programação (SANTOS et al., 2014). Jogos estão cada vez mais presentes como uma prática habitual no ensino e treinamento, sendo concebidos como uma atividade motivadora no processo de ensino-aprendizado (MONSALVE, WERNECK e LEITE, 2010). De maneira geral, um jogo é definido como um tipo de atividade conduzida em um contexto de realidade imaginada,

onde os participantes tentam alcançar ao menos uma meta, atuando de acordo com regras pré-estabelecidas (SAVI, 2011).

Neste sentido, não necessariamente se deve adotar jogos automatizados, pois os jogos educacionais são encontrados tanto em formatos não digitais (como tabuleiros e cartas) como em formatos digitais (disponíveis em computadores ou consoles). De acordo com Savi (2011), esses jogos possuem objetivos educacionais definidos, sendo projetados especificamente para ensinar determinados temas ou reforçar e apoiar a aprendizagem de habilidades. Um exemplo de jogo adotado no ensino de ES é o SimulES-W (MONSALVE, WERNECK e LEITE, 2010) que permite que um aluno assuma diferentes papéis num projeto de desenvolvimento de software.

Além dos jogos educacionais, destacam-se as ferramentas de simulação que, geralmente, simulam um contexto de uma fábrica de software, instituição de ensino e/ou empresa e seus respectivos participantes, onde cada estudante deve assumir um papel e desempenhar atribuições específicas nos seus ambientes (BESSA, CUNHA e FURTADO, 2012). Essas ferramentas permitem o desenvolvimento de experiências semelhantes as existentes no mundo real, reduzindo assim as lacunas existentes entre teoria e prática.

O uso de jogos e simuladores para treinar, aprender e executar atividades reais em ambientes realísticos pode melhorar o desempenho dos estudantes, pois possibilita a vivência de experiências de aprendizagem que não são fornecidas de forma teórica (MONSALVE, WERNECK e LEITE, 2010). Além disto, estes softwares educativos propiciam aos professores estimular os estudantes quanto à aprendizagem de forma lúdica.

A sua utilização tem demonstrado estes benefícios quando esses entram em contato com o objeto de estudo, facilitando assim o trabalho do professor, pois a dinâmica dos jogos permite ao aluno desenvolver habilidades a partir da compreensão de suas regras, bem como apreender os conceitos do conteúdo alvo do jogo e desenvolver aspectos afetivo-sociais (PRIKLADNICKI et al., 2009).

4.3.4 Discussão de Casos Práticos

Essa abordagem consiste basicamente na apresentação de relatos de experiências da indústria e na discussão dos problemas e soluções encontrados. A maioria dos relatos

de experiências se dá através de estudos de casos. De acordo com Marques, Quispe e Ochoa (2014) e Santos et al. (2014), grande parcela das abordagens de ensino de ES identificadas em seus mapeamentos sistemáticos tratam da discussão de estudos de casos.

As diretrizes curriculares da ACM/IEEE (2014) enfatizam que a discussão de estudos de caso permite incorporar elementos do mundo real ao ensino e auxilia na efetividade do ensino de conceitos. Esses estudos expõem projetos e sistemas para que os alunos possam analisar criticamente e reusar as soluções propostas (ACM/IEEE, 2014). Assim, os alunos desenvolvem habilidades em pensamento analítico lendo e discutindo problemas complexos da vida real (MARQUES, QUISPE e OCHOA, 2014).

4.3.5 Adaptação de Práticas da Indústria

Como a área da Computação não distingue bem entre as diferentes funções de desenvolvimento, a educação para os engenheiros de software é confundida com a educação para os programadores (SHAW, 2000). No entanto, a educação para os engenheiros de software deve preparar os estudantes de forma diferente para as diversas funções exercidas por estes na indústria, ajudando-os a se manterem atualizados em face às rápidas mudanças, e estabelecer competências que reflitam com precisão as habilidades requeridas para esses profissionais.

Nesse sentido, determinadas abordagens adaptaram práticas de capacitação da indústria para o ambiente acadêmico, a fim de melhor desenvolver essas habilidades, como a exemplo de Gnatz et al. (2003) e Garg e Varma (2008).

Gnatz et al. (2003) propuseram em um curso uma abordagem prática para o ensino de ES na *Technical University of Munich*, chamada *Software Engineering Project* (SEP). O principal objetivo da abordagem SEP é providenciar para os alunos uma experiência de vivenciar projetos de software semelhantes aos da indústria. Dessa forma, os autores adequaram determinadas práticas da indústria, como por exemplo, o uso de clientes e problemas reais, para o contexto de uma disciplina da área de ES.

A participação de profissionais da indústria em projetos acadêmicos é, atualmente, bastante adotada, como na abordagem de Chen e Chong (2011) em que os *stakeholders* do projeto participam de reuniões para fornecer instruções para os alunos. Santos et al. (2009) destacam que o cliente é o elemento mais importante e crítico para representar a

realidade do mercado, portanto, sua participação e envolvimento no projeto (mesmo acadêmico) é fundamental.

Nessa disciplina, os professores realizaram atividades de *coaching* que incluíram resolução de problemas técnicos, bem como a moderação de discussões sobre as atividades a serem realizadas e o planejamento dos próximos passos. Adicionalmente, os professores tiveram também que manter as equipes motivadas. Santos et al. (2009), em sua experiência prática de adaptação de fábrica de software para o contexto acadêmico, relataram dificuldades em mudar o comportamento dos professores para desempenhar um papel de *coach* e consultor. A fim de que o professor possa apoiar os alunos ao longo do desenvolvimento dos projetos, destacam que treinamentos de tutoria devem ser considerados na preparação das disciplinas.

Os alunos veteranos orientaram as equipes através do processo de desenvolvimento e realizaram a garantia da qualidade, sendo responsáveis por revisar todos os documentos produzidos ao longo do processo. A intenção por trás dessa divisão de responsabilidades foi garantir a independência entre os documentos produzidos pelos estudantes e a objetividade dos supervisores em analisar esses documentos.

Durante o uso dessa abordagem, destaca-se que os alunos precisaram gerenciar várias ferramentas, bem como algumas novas tecnologias, e foram obrigados a lidar com uma quantidade de código aumentando continuamente. Ao final do projeto, esse código tornou-se muito complexo para um único programador, e os estudantes perceberam a importância de modularização e da comunicação entre a equipe (GNATZ et al., 2003).

Semelhante à abordagem de Gnatz et al. (2003), a etapa de contextualização do modelo (ver Subseção 4.4.5) pretende permitir aos alunos realizarem um projeto de desenvolvimento mais próximo possível do real. Sendo assim, os professores devem realizar o papel de *coaching* e alunos veteranos podem orientar as equipes na realização de atividades técnicas, semelhante à prática de *mentoring*. Além disso, pretende-se envolver profissionais da indústria (ver Subseção 4.4.3) e utilizar ferramentas para apoiar o desenvolvimento das atividades técnicas do projeto.

Conforme destacado no capítulo introdutório deste trabalho, é prática recorrente nas empresas a capacitação de profissionais recém-formados. No entanto, retreinar os profissionais em conhecimentos e habilidades básicas aumenta os custos, tempo e complexidade aos programas de capacitação das empresas. Nesse contexto, Garg e Varma

(2008) propõem que a formação dos estudantes pode ser voltada para habilidades específicas de ES, conforme acontece na capacitação de profissionais da indústria.

Além disso, os autores destacam que a indústria investe muito dinheiro em treinamento, mas a principal dificuldade é encontrar treinadores habilitados que tragam experiências práticas de trabalho para compartilhar com os estudantes (GARG e VARMA, 2008). Da mesma forma, há escassez de professores qualificados na academia que tenham atuado na indústria. Esses professores podem ensinar aspectos teóricos, mas dificilmente poderão expressar habilidades intrínsecas ao processo de desenvolvimento.

O modelo proposto neste trabalho visa desenvolver competências e habilidades específicas de acordo com o papel que o aluno irá exercer no projeto (ver Subseção 4.4.1), assim como sugerido por Garg e Varma (2008). A fim de tratar a escassez de professores qualificados, pretende-se criar uma base de conhecimento, de acordo com as unidades de conhecimento da ES, com vídeos de palestras de profissionais que atuam em diferentes papéis (ver Seção 4.6).

4.4 SINTAXE DO MODELO

Em busca de potencializar os benefícios da adoção conjunta de abordagens práticas para o ensino de ES, esse modelo estabelece etapas bem definidas, dispostas em um ciclo iterativo a fim de atender diferentes perfis de aprendizagem. Esse ciclo é fundamentado na teoria de aprendizagem de Kolb (1984) e na metodologia iterativa de ensino proposta por Gary et al. (2013). Já os perfis de aprendizagem são baseados na classificação proposta por Fleming e Mills (1992).

A **Figura 7** apresenta as 6 (seis) etapas que compõem esse modelo de ensino e o fluxo de iteração entre elas. Esse modelo é centrado na leitura de artigos técnicos com relatos de experiências e videoaulas com fundamentação dos tópicos, combinando aprendizagem centrada em problema (PBL), discussão de casos práticos, uso de jogos, simuladores ou dinâmicas, realização de projeto prático e reflexão. Sendo assim, sua instanciação permite apresentar, praticar e aplicar os tópicos das unidades de ES a fim de proporcionar uma aprendizagem mais contextualizada para os alunos e, consequentemente, desenvolver as competências técnicas necessárias para os graduandos ingressarem no mercado de trabalho.

Figura 7 – Modelo Iterativo para o Ensino de Engenharia de Software

Fonte: Elaborado pelo autor.

Nas subseções a seguir são descritas cada uma das etapas desse modelo, com destaque para a aplicação das abordagens de ensino selecionadas e para a adaptação das práticas de capacitação da indústria. Para contextualizar essa descrição, selecionou-se a unidade de conhecimento Engenharia de Requisitos (ER) como exemplo.

4.4.1 Etapa de Iniciação

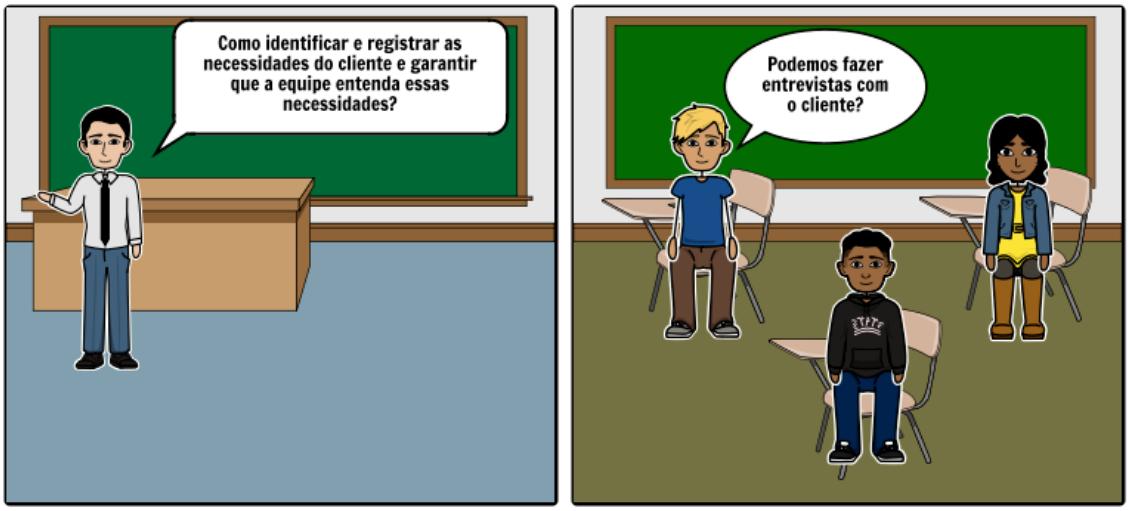
Inicialmente, o professor deve selecionar uma unidade de conhecimento que será objeto de estudo. Adicionalmente, ao se definir à unidade de conhecimento, se faz necessário identificar quais competências pretende-se desenvolver nos alunos. No exemplo de uso da unidade de Engenharia de Requisitos, as competências a serem desenvolvidas são:

- Elicitação de Requisitos de Software;
- Análise de Requisitos de Software;
- Especificação de Requisitos de Software;
- Verificação e Validação de Requisitos de Software;
- Gerenciamento de Requisitos dos Processos e dos Produtos de Software.

Para as demais unidades, os professores devem consultar o SWECOM (IEEE COMPUTER SOCIETY, 2014A). Essas competências irão determinar as atividades técnicas a serem desenvolvidas na etapa de Prática (ver Subseção 4.4.4).

Posteriormente, o estudo de cada unidade de conhecimento começa a partir da identificação de um problema relacionado ao projeto a ser desenvolvido. Um exemplo ilustrado é apresentado na **Figura 8**.

Figura 8 – Exemplo de um Problema na Etapa de Iniciação



Fonte: Elaborado pelo autor.

Essa etapa é fortemente baseada na abordagem PBL (ver Subseção 4.3.2) e na primeira etapa do processo de *coaching*, Indagação, na qual realizam-se perguntas significativas (ver Subseção 2.5.2). A realização de perguntas significativas e o estabelecimento de metas desafiadoras constituem a principal intervenção do professor na sua atuação como *coach*.

Após selecionar a unidade de conhecimento foco da aplicação do modelo, especificar as competências que serão desenvolvidas, e identificar um problema estimulante e desafiador para introduzir a unidade, o professor pode avançar para as etapas de Preparação e/ou Discussão.

4.4.2 Etapa de Preparação

Em paralelo a todas as etapas do ciclo iterativo apresentado na Figura 7, deve ocorrer a Preparação dos alunos. Essa etapa consiste no acompanhamento de videoaulas relacionadas às unidades e na realização da leitura de artigos com relatos de casos práticos da indústria a fim de fundamentar, exemplificar e criar compreensão de como estes tópicos são aplicados. Assim, ocorrem os estímulos textual (artigos) e visual (videoaulas). Essa preparação pode ser feita tanto em sala de aula, auxiliada pelo professor, quanto em

casa (extra-classe), por iniciativa dos alunos. Para tomar esta decisão, o professor deve considerar o tempo disponível para as atividades da disciplina.

Para a unidade de Engenharia de Requisitos, sugere-se que o professor busque artigos com relatos de experiências nos anais do Workshop em Engenharia de Requisitos (WER) da *Conferencia Iberoamericana de Software Engineering* (CIbSE) e no Simpósio Brasileiro de Engenharia de Software (SBES) do Congresso Brasileiro de Software: Teoria e Prática (CBSoft). Quanto às videoaulas, recomenda-se as disponibilizadas no portal Videoaula@RNP (*http://www.videoaula.rnp.br/*) que armazena o conteúdo produzido por instituições clientes da Rede Nacional de Ensino e Pesquisa (RNP) ou ainda as diversas disponibilizadas nos canais do *YouTube* (*https://www.youtube.com*). A **Figura 9**Erro! Fonte de referência não encontrada. apresenta um exemplo ilustrado dessa etapa.

Figura 9 – Exemplo da Indicação de um Vídeo e Artigo na Etapa de Preparação

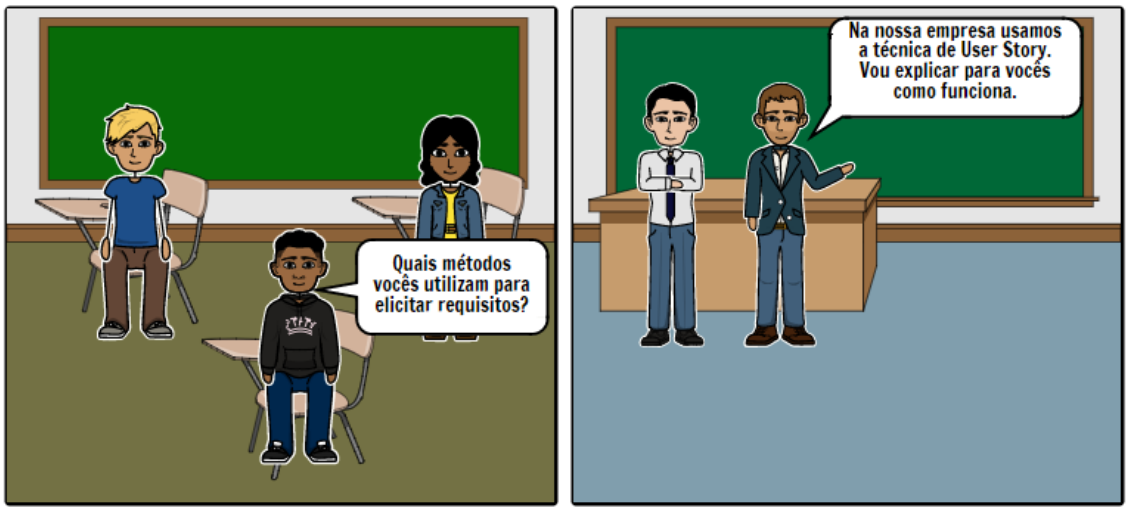
Fonte: Elaborado pelo autor.

O professor também pode criar seu próprio material, seja artigo ou videoaula, ou disponibilizar algum que já tenha conhecimento e que aborde o conteúdo estudado. O objetivo principal é fornecer recursos para que o aluno possa construir sua fundamentação teórica através do estudo de conceitos e definições na unidade de conhecimento estudada. Essa fundamentação é disponibilizada, geralmente, nas videoaulas. Já os artigos com relatos de experiências podem apresentar um caso prático que seja selecionado como foco da etapa de Discussão.

4.4.3 Etapa de Discussão

Após a etapa de Iniciação, os alunos devem realizar uma Discussão com um profissional especialista na unidade de conhecimento, como um aluno egresso da instituição que atue na indústria como Analista de Requisitos, por exemplo. Diante de uma eventual impossibilidade da visita do profissional à sala de aula, a discussão pode se dar a partir de uma palestra em vídeo, videoconferência ou ainda a partir de uma visita técnica a uma empresa de software. Outra sugestão seria convidar alunos concluintes da disciplina ou egressos que defenderam trabalhos de conclusão com tema relacionado à unidade. Uma discussão que pode ocorrer nesta etapa é exemplificada na **Figura 10**.

Figura 10 – Exemplo de uma Interação na Etapa de Discussão



Fonte: Elaborado pelo autor.

A discussão possibilita aos alunos uma visão mais próxima de como determinados tópicos são aplicados na prática profissional do engenheiro de software, além de permitir uma análise crítica e reuso das soluções propostas, conforme benefícios citados na abordagem de discussão de casos práticos (ver Subseção 4.3.4).

Essa discussão entre alunos, professor e profissional convidado deve girar em torno do problema levantado na fase de Iniciação ou identificado na leitura de artigos. Outras possibilidades seriam o profissional descrever problemas enfrentados no dia-a-dia da sua empresa ou os alunos o questionarem a partir de suas dúvidas. O importante é que o professor aja como mediador da discussão durante essa etapa a fim de gerenciar o tempo do convidado e garantir que não ocorram desvios do foco da discussão.

Adicionalmente, o profissional convidado pode exercer, de acordo com a sua disponibilidade, algumas atividades relacionadas à prática de *mentoring*. Por envolver aconselhamento e inspiração, recomenda-se que esse profissional tenha no mínimo um conhecimento básico do papel de mentor e do processo de mentoria, além de saber orientar outras pessoas nos métodos, ferramentas e técnicas que utiliza durante a realização das suas atividades.

Após a realização da discussão, espera-se que os alunos sanem a maioria das suas dúvidas e obtenham sugestões de métodos, ferramentas e técnicas aplicáveis na etapa de Contextualização. Antes de aplicar esse conhecimento, faz-se necessário que o aluno o compreenda através da realização de atividades lúdicas na etapa de Prática.

4.4.4 Etapa de Prática

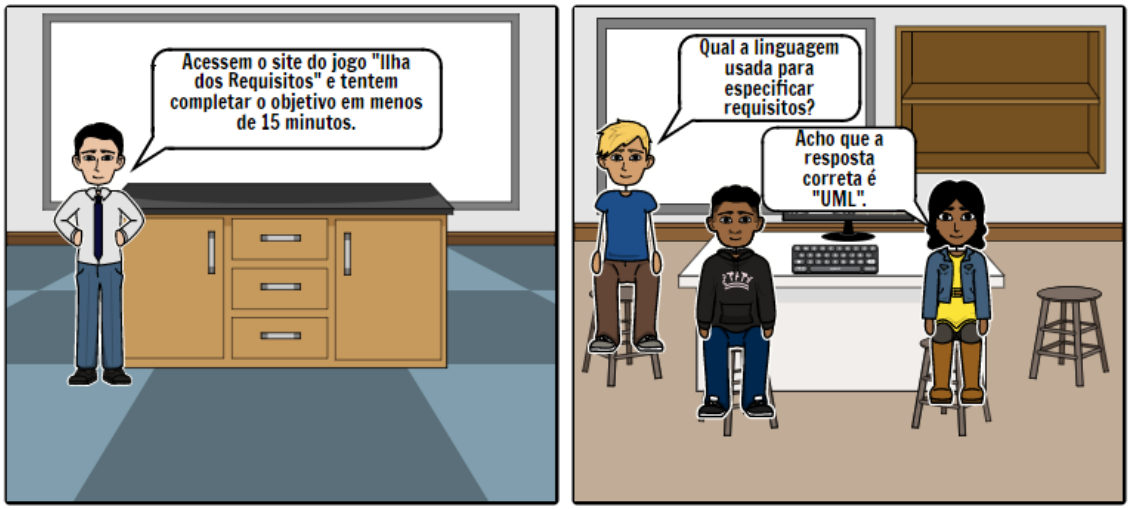
Após discutir sobre a unidade de conhecimento, os alunos devem colocar os conhecimentos obtidos em Prática através do uso de jogos ou simuladores (ver Subseção 4.3.3). Essa etapa permite aos alunos internalizarem determinadas habilidades a partir de atividades práticas, pois esses softwares educacionais possibilitam a vivência de experiências de aprendizagem que não são possíveis através de leituras e discussões teóricas. Adicionalmente, permite aos professores motivar e estimular os alunos quanto à aprendizagem. Aos alunos, possibilita desenvolver habilidades e apreender conceitos relacionados ao conteúdo do jogo, além de desenvolver aspectos de interação e comunicação com os seus pares.

Existem vários jogos voltados ao ensino de Engenharia de Requisitos publicamente disponíveis. Recomenda-se o jogo denominado “Ilha dos Requisitos” (THIRY, ZOUCAS e GONÇALVES, 2010). Nesse jogo, o aluno/jogador se torna um sobrevivente da queda de um avião numa ilha e deve resolver desafios referentes à ER para conseguir escapar dela. Esse jogo é single-player e está disponível em *http://www.incremental.com.br/ilhadosrequisitos/*. Um exemplo ilustrado do uso desse jogo é apresentado na **Figura 11**.

Além dos jogos educacionais, destaca-se o uso recente na ES da técnica de *Gamification* que consiste na aplicação de elementos e estratégias de jogos virtuais no ensino (DETERDING et al., 2011). Essa técnica pode ser aplicada nessa etapa de maneira combinada com o uso de jogo. Por exemplo, pode-se gerar um *ranking* a partir do desempenho obtido pelos alunos no decorrer do jogo. Para o aluno que obtiver a melhor

classificação da turma, ou para os membros que obtiverem a melhor média da equipe, o professor pode recompensá-lo(s) com um exemplo de Caso de Uso modelado e descrito, por exemplo.

Figura 11 – Exemplo de Uso de um Jogo na Etapa de Prática



Fonte: Elaborado pelo autor.

Nessa etapa, o professor também pode adaptar e aplicar práticas de capacitação da indústria. Assim, uma recomendação seria realizar dinâmicas relacionadas à unidade de conhecimento ou mesmo realizar um *workshop* de treinamento sobre o uso de ferramentas de apoio. Ambas as práticas permitem a interação entre os alunos e o facilitador, que pode ser o professor ou um convidado (outro professor, um aluno egresso ou profissional convidado).

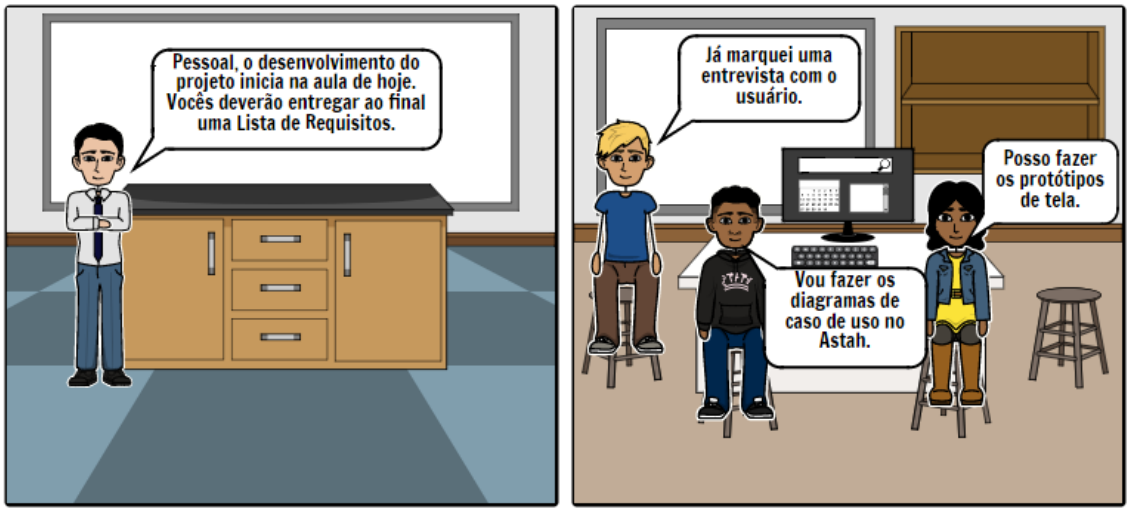
Após discutir e colocar em prática os conceitos internalizados, os alunos são considerados aptos a avançarem à etapa de Contextualização a partir da realização de um projeto prático de desenvolvimento de software.

4.4.5 Etapa de Contextualização

Nesta etapa, finalmente os alunos têm a oportunidade de integrar as habilidades adquiridas a partir da realização de um projeto prático de desenvolvimento de software. Esse projeto busca a Contextualização da aplicação dos tópicos através da adoção de técnicas imersivas de ensino/aprendizagem, como o uso de clientes/problemas reais, definição de prazos (cronograma), divisão de papéis e responsabilidades pelos alunos.

O objetivo principal é fornecer ao aluno a experiência de um projeto de desenvolvimento similar ao da indústria. Assim, os alunos poderão aprender técnicas diversas, como definição do projeto, gerenciamento, programação, validação, análise, entrevistas de usuários, documentação, dentre outras. Um exemplo de realização de projeto prático é ilustrado na **Figura 12**.

Figura 12 – Exemplo de um Projeto Prático na Etapa de Contextualização



Fonte: Elaborado pelo autor.

Além da parte técnica, essa experiência deve permitir desenvolver habilidades de avaliação de potenciais soluções, negociação com os clientes, trabalho em grupo, comunicação e relações interpessoais. No entanto, é importante destacar que o professor deve considerar as limitações do ambiente acadêmico na definição do tema, escopo, prazo e qualidade esperada para o projeto a ser desenvolvido pelos alunos. Assim, pode ser necessário selecionar quais competências serão priorizadas durante o projeto em detrimento de outras. A escolha do tema fica a critério do professor, que deve discuti-lo e procurar obter o consenso dos alunos.

Nesse sentido, o professor pode incorporar na etapa de Contextualização somente atividades relacionadas à unidade de conhecimento foco da aplicação do modelo ou integrar estas com atividades de outras unidades. Por exemplo, pode integrar as atividades de Engenharia de Requisitos com atividades das unidades de Desenvolvimento de Software e Verificação e Validação.

Sendo assim, para atender às competências da unidade de Engenharia de Requisitos, o SWECOM (IEEE COMPUTER SOCIETY, 2014A) propõe a realização das atividades listadas na **Tabela 8**.

Tabela 8 – Competências e Atividades da Unidade de Engenharia de Requisitos

Competência	**Atividades**
Elicitação de Requisitos de Software	• Identificar os *stakeholders* (partes interessadas) para elicitação de requisitos; • Envolver as partes interessadas na elicitação de requisitos; • Usar métodos apropriados para capturar requisitos; • Negociar conflitos entre as partes interessadas durante a elicitação.
Análise de Requisitos de Software	• Utilizar técnicas de análise de domínio apropriadas; • Realizar análise de viabilidade dos requisitos e propriedades emergentes.
Especificação de Requisitos de Software	• Utilizar notações apropriadas para descrever requisitos.
Verificação e Validação de Requisitos de Software	• Verificar os requisitos quanto à acurácia, falta de ambiguidade, integridade, consistência, rastreabilidade e outros atributos desejados; • Construir e analisar protótipos; • Negociar conflitos entre *stakeholders* durante a verificação.
Gerenciamento de Requisitos dos Processos e dos Produtos de Software	• Usar métodos apropriados para o gerenciamento de requisitos, incluindo gerenciamento de configuração.

Fonte: Adaptado do SWECOM (IEEE COMPUTER SOCIETY, 2014A).

Quanto à execução dessas atividades, a ACM/IEEE (2013) destaca, através das suas pesquisas sobre o ensino de ES, que há evidências crescentes de que os estudantes aprendem a aplicar os princípios de ES de forma mais eficaz através do uso de abordagens iterativas. Essas abordagens permitem aos alunos realizarem suas atividades em um ciclo de desenvolvimento, avaliar o seu trabalho ao final do ciclo, em seguida, aplicar o conhecimento adquirido em um próximo ciclo de desenvolvimento. Dessa forma, recomenda-se que os professores trabalhem com modelos de ciclo de vida ágeis, como o Scrum e XP (KNIBERG, 2008) que providenciam essas oportunidades aos alunos.

Na Contextualização, o professor pode realizar diversas etapas do processo de *coaching*. Por exemplo: fazer indagações para auxiliar os alunos na escolha de métodos e técnicas; esclarecer objetivos e metas do projeto prático; auxiliar na definição dos papéis e responsabilidades da equipe bem como na definição das tarefas e do cronograma; solicitar e dar *feedback* durante a realização das atividades.

Ao finalizar as atividades práticas do projeto na etapa de Contextualização, solicita-se que os alunos analisem o processo seguido durante essa etapa e realizem uma reflexão sobre os resultados obtidos.

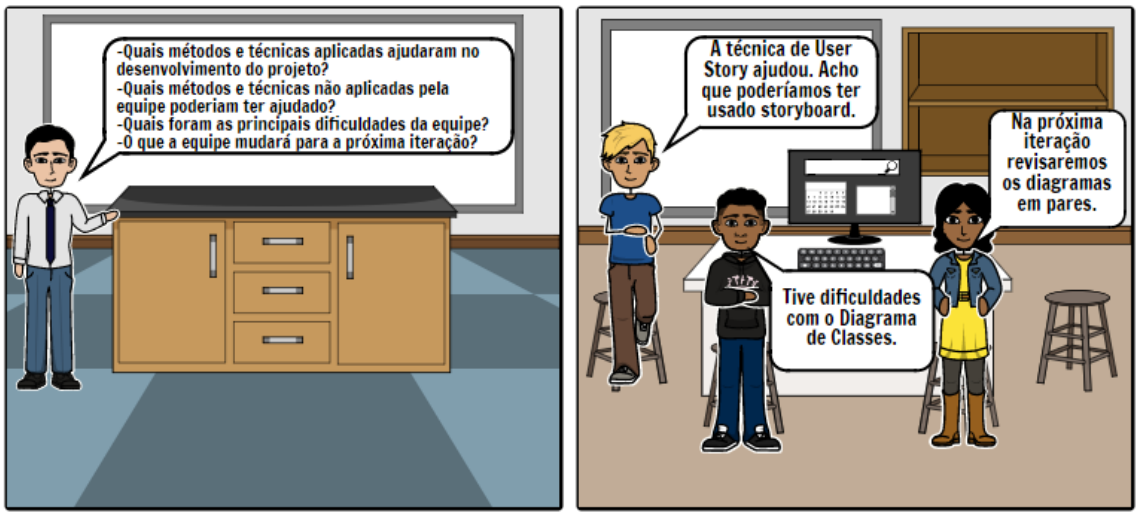
4.4.6 Etapa de Reflexão

Ao final da etapa de Contextualização, os alunos apresentam os resultados do projeto prático para o professor, a turma, profissional convidado e demais *stakeholders*. Após essa apresentação, devem realizar uma Reflexão sobre essa rápida experiência de aprendizagem, e se envolver em grupos de discussão sobre as lições aprendidas.

O profissional que participou da Discussão pode ser convidado novamente para participar dessa etapa. Assim, poderia concluir o processo de *mentoring* através da visualização dos resultados da etapa de Contextualização e da análise de possíveis mudanças no comportamento de seus mentorandos.

Nessa etapa, basicamente, os alunos devem responder a 4 (quatro) perguntas, baseadas na cerimônia *Sprint Retrospective* do Scrum (KNIBERG, 2008), conforme ilustra a **Figura 13**Erro! Fonte de referência não encontrada..

Figura 13 – Exemplo de uma Reunião na Etapa de Reflexão



Fonte: Elaborado pelo autor.

Essas reflexões devem ser armazenadas pelo professor na base de dados do sistema do modelo a fim de servir de um repositório de experiências que pode auxiliar futuros alunos que irão estudar a mesma unidade de conhecimento. O ciclo iterativo do modelo apresentado na Figura 7 deve ser executado para cada unidade de conhecimento

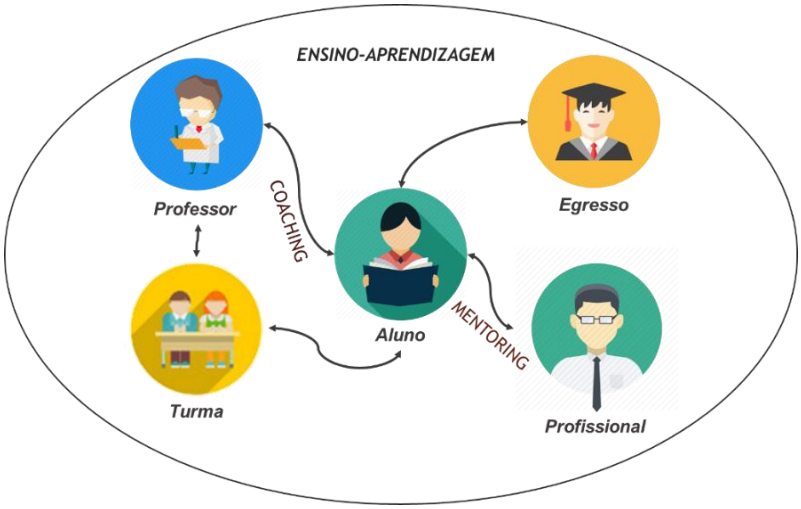
da ES que o professor deseja ministrar. Caso o professor opte por ministrar mais de uma unidade, o ciclo iterativo deve começar novamente desde a fase de Iniciação.

4.5 SISTEMA SOCIAL

A interação entre alunos e professores em cada modelo de ensino é vista de modo análogo a uma mini sociedade (JOYCE e WEIL, 1972). Dessa forma, um sistema social deve especificar as funções interativas e relações entre o professor e o aluno e quais comportamentos são esperados de ambos para que o objetivo do modelo seja atingido. Nesse contexto, estratégias motivacionais para envolver os alunos também devem ser aplicadas.

Hall e Fagan (1956) definem um sistema simplesmente como um "conjunto de objetos juntos com relacionamentos entre esses objetos e seus atributos". No modelo proposto, o sistema social engloba os seguintes objetos: o professor; o aluno; a turma; o profissional convidado ou o aluno egresso; o processo de ensino-aprendizagem. Essas entidades e seus relacionamentos são representados na **Figura 14**.

Figura 14 – Sistema Social do Modelo



Fonte: Elaborado pelo autor.

As características de cada uma dessas entidades, seus relacionamentos e papéis no sistema social são apresentadas nas subseções a seguir.

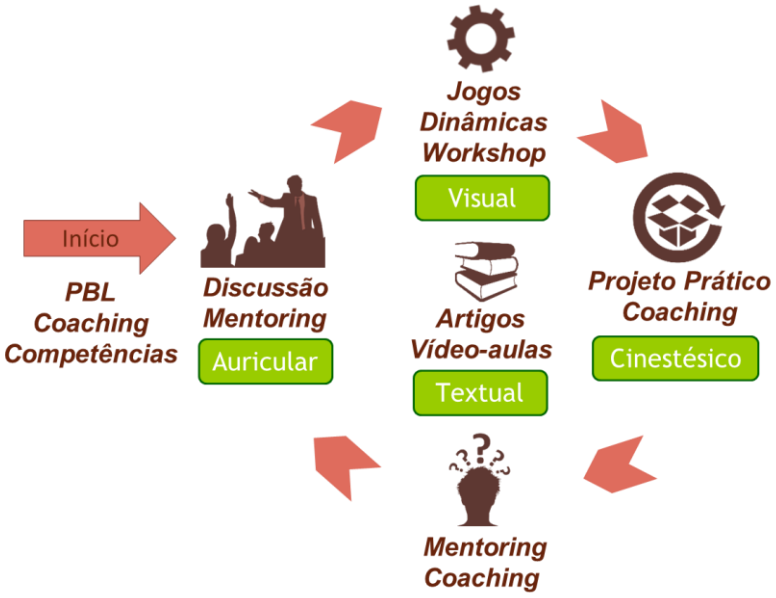
4.5.1 O Aluno

Nesse sistema, o aluno é o centro do processo de ensino-aprendizagem. No entanto, isso acarreta no aumento da sua responsabilidade, pois esse deve assumir um

perfil participativo e ter voz ativa. Deve buscar resolver os problemas propostos pelo professor de forma criativa e original através da aplicação de conhecimento na realização das atividades práticas.

Para tal, deve solicitar orientação ao profissional convidado a respeito de métodos, ferramentas e técnicas que pode vir a utilizar nessas atividades. Destaca-se que cada aluno possui um perfil de aprendizagem diferente, o qual deverá ser identificado e considerado pelo professor no processo de ensino-aprendizagem. As técnicas de ensino adotadas em cada etapa do modelo, a fim de atender às necessidades de aprendizagem dos diferentes perfis de alunos, são destacadas na **Figura 15**.

Figura 15 – Técnicas e Estímulos de Ensino Adotadas no Modelo



Fonte: Elaborado pelo autor.

No entanto, Santos (2005) destaca que, em decorrência das pesquisas realizadas, leituras, experiências sociais e afins, o professor incorpora de certa forma um ou mais aspectos dos referenciais teóricos analisados anteriormente em suas práticas docentes, muitas das quais são derivadas de como ele foi educado durante sua vida escolar. Dessa forma, descreve-se a seguir o papel a ser seguido pelo professor que adotar este modelo de ensino.

4.5.2 O Professor

O professor deve projetar situações de aprendizagem e atividades que estimulem e desafiem os alunos. Assim, deve assumir um perfil de facilitador do processo de ensino-

aprendizagem. No entanto, esse perfil exige uma grande responsabilidade sobre os professores no processo de guiar os alunos para encontrarem a "solução", pois devem filtrar o conhecimento de diferentes fontes, e organizá-lo para apresentar aos alunos. Além disso, devem determinar a quantidade correta de orientação e apoio que irão fornecer às equipes dos projetos a fim de permitir a aprendizagem sem causar grandes impactos nos projetos, desviando os alunos de encontrar por si mesmos a "solução".

Nesse sentido, sugere-se que o professor incorpore, adicionalmente, o perfil de *coach*. Essa prática sugere a definição de objetivos estimulantes; o estabelecimento de metas desafiadoras; o respeito às diferenças individuais; o fornecimento/recebimento de *feedback*; e o apoio na realização das atividades. Através do seguimento dessa prática, o professor poderá estimular e apoiar o aluno a desenvolver as competências técnicas esperadas. Outra forma de estimular os alunos, incorporada ao modelo, é a adoção de softwares educativos (como jogos e simuladores) e a sugestão da aplicação de estratégias de gamificação.

No entanto, todas essas estratégias podem não ser suficientes para motivar os alunos. Dessa forma, este modelo sugere a inserção de um terceiro elemento nesse sistema social: o profissional. Espera-se que ouvir e interagir com um aluno egresso de cursos da área da Computação, falando sobre sua carreira profissional em Engenharia de Software, possa incentivar e engajar os alunos no processo de ensino-aprendizagem.

4.5.3 O Profissional

A fim de aconselhar, inspirar e motivar os alunos, sugere-se que o professor convide um aluno egresso da instituição que atue profissionalmente na área relacionada à unidade de conhecimento ministrada. Esse profissional pode atender a esses objetivos através do relato de experiências e problemas recorrentes que ele enfrenta no dia-a-dia da sua profissão e da apresentação das soluções técnicas adotadas nestas situações.

Nesse sentido, sugere-se que o profissional realize um processo semelhante ao *mentoring*. Esse processo sugere o uso de práticas como metáforas, analogias, histórias de personagens de sucesso, exemplos, demonstrações, relatos de experiência, dentre outras. Através do seguimento dessas práticas, o profissional poderá instruir e orientar os alunos na escolha e no uso de métodos, ferramentas e técnicas utilizadas para realizar atividades técnicas relacionadas à unidade foco da aplicação do modelo.

Ciente das dificuldades em trazer um profissional da indústria até a sala de aula, o professor poderia fazer o convite a um aluno concluinte da disciplina, ou a um egresso que tenha realizado iniciação científica ou trabalho de conclusão com tema relacionado à unidade. Em todos os casos, é necessário que o professor passe as instruções necessárias ao profissional/aluno egresso a respeito do papel que esse deve desempenhar durante as etapas de Discussão e Reflexão, usando como base a documentação do modelo.

4.5.4 O Processo de Ensino-Aprendizagem

No que diz respeito ao processo de ensino-aprendizagem, Mizukami (1986) destaca que esse será influenciado tanto pelo caráter individual do professor quanto por como ele se relaciona com o caráter pessoal do aluno. Nesse sentido, os objetivos educacionais devem acompanhar o desenvolvimento psicológico do aluno. Os conteúdos da unidade devem ser selecionados a partir dos interesses dos alunos. Logicamente, esses conteúdos devem se relacionar com as competências a serem desenvolvidas, conforme destacado na Subseção 4.4.1.

O processo de ensino-aprendizagem está diretamente ligado com o seguimento da sintaxe do modelo (ver Seção 4.4) pelas entidades envolvidas. Inicialmente, o professor deve identificar problemas técnicos relacionados à unidade de conhecimento a fim de elaborar indagações pertinentes aos alunos. Na fase de Discussão, o profissional convidado deve atuar como um mentor, motivando, instigando e apresentando experiências e soluções. O aluno, por sua vez, deve aproveitar a oportunidade para realizar questionamentos pertinentes e absorver algum nível de conhecimento teórico. Em seguida, na fase de Prática, o aluno poderá aplicar o conhecimento obtido e então será necessário que o professor adote alguma abordagem motivadora, como jogos e dinâmicas.

A fase de Contextualização é considerada crítica para o processo de ensino-aprendizagem, pois depende altamente da dedicação e empenho do aluno. Muitas das atividades dessa etapa são realizadas fora de sala de aula. Sendo assim, é necessário que o professor acompanhe e solicite *feedback* do andamento do projeto. Por outro lado, o aluno deve se comprometer com a execução das atividades e fornecer *feedback* constante ao professor. Em caso de dificuldades, principalmente técnicas, os alunos podem e devem solicitar apoio ao docente.

Na última fase, Reflexão, sugere-se que o profissional retorne à sala de aula tanto para observar o resultado do trabalho dos alunos quanto para identificar possíveis

mudanças no comportamento desses advindas da sua intervenção. Adicionalmente, o professor deve explicitar, de forma reflexiva, os resultados obtidos e as mudanças identificadas nos alunos ao final de um ciclo iterativo.

Por fim, destaca-se que esse processo de ensino-aprendizado deve ocorrer principalmente em sala de aula, mas não se limitando a esse ambiente. A sala de aula é o ambiente de convívio das entidades do sistema social, exercendo influência sob esses.

4.5.5 A Sala de Aula

Santos (2005) destaca que a escola (ou instituição de ensino) está intimamente ligada ao processo social, sendo ao mesmo tempo agente influenciador e influenciado por esse. Nesse contexto, o ambiente social da sala de aula envolve aspectos do apoio do professor (emocional e acadêmico), bem como a interação social estudantil promovida por ele e o respeito mútuo (STEWART, 2014). As salas de aula, como um ambiente social positivo, tendem a promover o sentimento de pertencimento a um grupo, satisfação, entusiasmo e respeito entre os alunos.

De acordo com Stewart (2014), o apoio emocional refere-se à percepção do aluno de que o professor se preocupa com ele e gosta dele como pessoa, enquanto o apoio acadêmico refere-se à percepção de que o professor se preocupa com o quanto ele aprende e o quanto quer ajudá-lo a aprender. Quando os alunos percebem essas características em seus professores, eles são mais propensos a se envolverem em sala de aula e a se empenharem mais academicamente.

A interação social ocorre à medida em que os alunos percebem que os professores encorajam a cooperação e a colaboração, fazendo com que os alunos compartilhem ideias, trabalhem juntos em atividades em pequenos grupos, se envolvam na busca de ajuda informal ou ajudem durante a realização de um trabalho individual.

Por fim, o respeito mútuo se estabelece à medida em que os alunos percebem os professores incentivando o respeito entre colegas de classe. Um foco no respeito mútuo deve ajudar a criar um ambiente onde os alunos se comunicam positivamente uns com os outros e, consequentemente, dediquem mais recursos cognitivos para se envolverem nas tarefas desenvolvidas em sala de aula.

4.6 APLICAÇÃO DO MODELO

De acordo com Pateliya (2013), a adoção de modelos de ensino ajuda o professor a ter uma ampla variedade de abordagens para a criação de um ambiente interativo adequado para a aprendizagem. Esses modelos especificam situações ambientais particulares que fazem com que o estudante possa interagir de tal forma que ocorra uma mudança específica no seu comportamento (JOYCE e WEIL, 1972).

A partir dessas premissas, os requisitos e materiais de apoio, bem como o processo de instanciação e de desenvolvimento de competências nesse modelo são descritos nas subseções a seguir.

4.6.1 Requisitos e Materiais de Apoio

Os requisitos necessários para que os professores implementem este modelo são:

- O professor deve ter formação em nível de graduação ou pós-graduação na área de Computação;
- O professor deve ter interesse no uso de abordagens práticas de ensino focadas no aluno;
- O professor deve possuir um perfil de facilitador do processo de ensino-aprendizagem;
- O professor deve buscar identificar e considerar os diferentes tipos de aprendizagens dos alunos no processo de ensino-aprendizagem.

Além desses, é desejável que o professor possua experiência na indústria de software. Essa experiência tende a facilitar a aplicação do modelo, pois o professor poderá contextualizar melhor o ensino das unidades de ES e, além disso, poderá ter um conhecimento prévio das práticas de capacitação adotadas no treinamento de profissionais na indústria.

Quanto aos materiais de apoio, conforme destacado na Seção 4.1, são compostos pela especificação do modelo, exemplo de uso e sistema *web*. Os exemplos contidos na especificação e no exemplo de uso são focados em Engenharia de Requisitos. Selecionar o material instrucional das demais unidades é uma das competências esperadas de um professor. Para tal, poderá acessar os materiais compartilhados das demais unidades na base de dados do sistema do modelo.

Além dos materiais, para aplicar o modelo se faz necessário o uso de equipamentos para instalar os jogos e simuladores, executar as videoaulas e realizar o projeto prático da disciplina. Esses requisitos incluem, mas não se limitam a, equipamentos de hardware, software e infraestrutura de redes. Por exemplo, o jogo "Ilha dos Requisitos" (THIRY, ZOUCAS e GONÇALVES, 2010) é executado no navegador *web*, sendo necessário um computador com acesso à Internet. Da mesma maneira, a maioria das videoaulas encontram-se disponível na *web*.

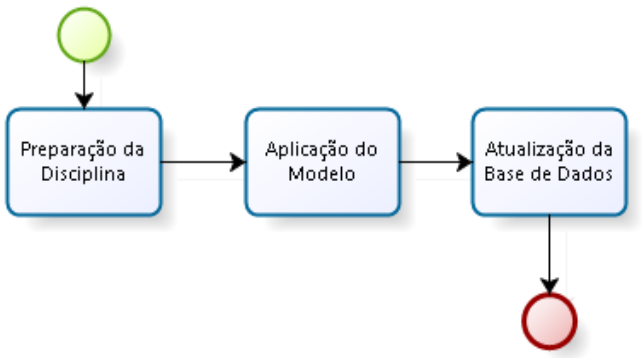
Adicionalmente, para exibir as videoaulas, o professor necessita de um computador, projetor e caixa de som. Para possibilitar o desenvolvimento de software resultante do projeto prático, o professor deve ter disponível no mínimo um laboratório equipado com computadores, com os softwares necessários instalados e acesso à Internet. Por fim, destaca-se a necessidade do professor providenciar o material utilizado nas práticas de capacitação da indústria a serem adotadas, como dinâmicas e *workshops*.

4.6.2 Instanciação do Modelo

A instanciação do modelo é flexível, sendo possível adotá-lo tanto para o ensino de uma unidade de conhecimento específica quanto para uma disciplina da área de ES completa. Apesar da possibilidade de aplicar o modelo em duas ou mais disciplinas concomitantes, que possuam conteúdos relacionados como, por exemplo, Análise e Projeto e Desenvolvimento Web, não é recomendado fazê-lo, pois exige um alto grau de sincronia entre o cronograma e as atividades destas disciplinas.

Assim, para uma disciplina específica, a sua instanciação deverá ocorrer em 3 (três) fases, conforme representado na **Figura 16**.

Figura 16 – Fases da Instanciação do Modelo



Fonte: Elaborado pelo autor.

Inicialmente, o professor deve realizar a preparação da disciplina para aplicação do modelo. Assim, ele poderá definir a(s) unidade(s) de conhecimento que abordará nessa aplicação e selecionar as competências a serem desenvolvidas. Essa preparação deve incluir a seleção do material instrucional, como os artigos, videoaulas, dinâmicas, jogos e planejamento do projeto prático a ser desenvolvido. Em complemento, o professor deve convidar um profissional que atue em áreas relacionadas à(s) unidade(s) e planejar as atividades de *mentoring* e *coaching*. Após realizar essas atividades, o professor pode criar uma conta no sistema do modelo e cadastrar a disciplina que será alvo da aplicação do modelo.

Na segunda fase, inicia-se a aplicação do modelo em sala de aula, onde o professor deverá seguir as etapas destacadas na Seção 4.4. Nessa fase, o modelo pode ser aplicado seguindo o cronograma de aulas da disciplina, onde a cada período (geralmente de 50 minutos), deve-se realizar uma etapa do modelo, com exceção da etapa de Contextualização que deve durar de acordo com a complexidade e tempo necessário para realizar as atividades práticas do Projeto. O professor terá acesso à base de dados do modelo, com os artigos técnicos, videoaulas, jogos, propostas de *workshops* e dinâmicas, atividades do projeto, bem como um formulário para registro das reflexões.

Por fim, na fase de atualização da base de dados, o professor pode colaborar com a base de dados do modelo a partir da submissão do seu próprio material usado na aplicação do modelo. Assim, espera-se que a base do modelo se mantenha atualizada e cresça de maneira colaborativa. Todo material submetido pelos professores estará visível para os demais usuários do sistema e disponível sob licença *Creative Commons*.

4.6.3 Desenvolvimento de Competências

O modelo de ensino proposto busca desenvolver conhecimento nos alunos através da leitura de artigos, do acompanhamento de videoaulas e de discussões relacionadas às unidades de conhecimento da ES. A partir dessas práticas, o modelo contempla o nível "Conhecer", onde o aluno deve ter a capacidade de lembrar do conteúdo discutido nas etapas iniciais do modelo, Iniciação, Preparação e Discussão.

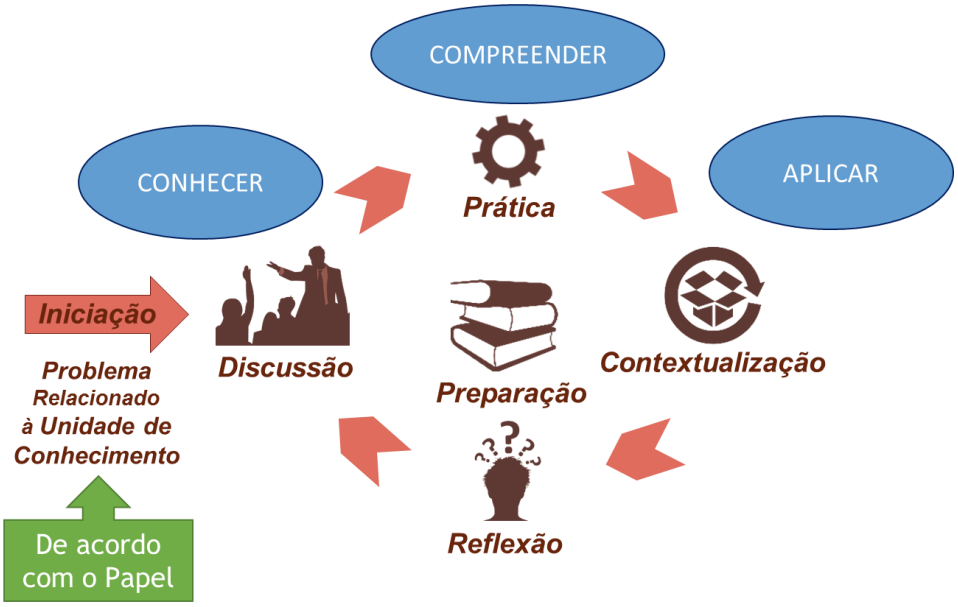
Esse conhecimento prévio pode ser transferido para outras situações na próxima etapa do modelo, denominada Prática, que consiste em uma breve experiência prática. Através do uso de jogos e simuladores ou da realização de dinâmicas e *workshops* de ferramentas de apoio, os alunos têm a oportunidade de desenvolver experiências

semelhantes às existentes no mundo real, reduzindo assim algumas lacunas existentes entre teoria e prática. Tais experiências de aprendizagem permitem ao modelo contemplar o nível "Compreender", onde o aluno deve ter a capacidade de entender o significado do conteúdo estudado.

Em seguida, na etapa de Contextualização, os alunos podem aplicar o conhecimento adquirido durante a execução do projeto prático. Dessa forma, o modelo contempla o nível "Aplicar", onde o aluno deve ter a capacidade de usar o conhecimento adquirido em situações práticas. Por exemplo, aplicando métodos, conceitos e técnicas para resolver problemas do projeto que requerem o conhecimento obtido nas etapas anteriores do modelo.

A **Figura 17** ilustra esses 3 (três) níveis de aplicação do conhecimento (BLOOM, 1956) no modelo, situando-os de acordo com as etapas.

Figura 17 – Níveis de Aplicação do Conhecimento no Modelo



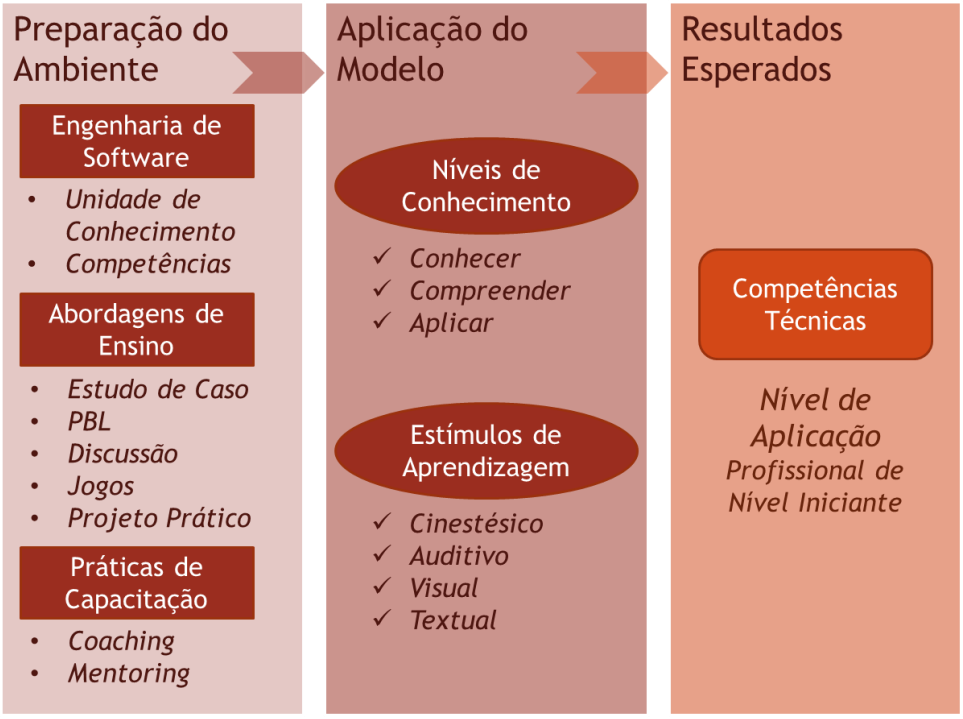
Fonte: Elaborado pelo autor.

Nessa figura, destaca-se o uso implícito de mais uma abordagem da indústria. Ao focar em uma unidade de conhecimento específica, o modelo proposto acaba por desenvolver competências e habilidades específicas de acordo com o papel que o aluno irá exercer no projeto. Por exemplo, se o aluno for atuar como Analista de Requisitos deve focar principalmente nas unidades de conhecimento relacionadas a esse papel, ou seja, Engenharia de Requisitos e Verificação e Validação.

Essa abordagem é semelhante à adotada durante o treinamento de profissionais na indústria que busca desenvolver competências relacionadas ao papel desempenhado pelo profissional. No caso específico da área de ES, essas competências estão relacionadas com a capacidade que o estudante deve adquirir para realizar atividades de desenvolvimento de software (NUNES, 2015).

A **Figura 18** apresenta as fases do processo de desenvolvimento de competências no modelo.

Figura 18 – Desenvolvimento de Competências no Modelo



Fonte: Elaborado pelo autor.

Inicialmente, o professor prepara o ambiente, determinando a unidade de conhecimento foco da aplicação do modelo, bem como as competências relacionadas a essa, selecionando-as no SWECOM (IEEE COMPUTER SOCIETY, 2014A). Em seguida, o professor prepara o material instrucional referente à aplicação das abordagens de ensino. Adicionalmente, o professor deve se preparar para aplicar técnicas de *coaching* e identificar e convidar um profissional da indústria para realizar *mentoring* dos alunos.

Após preparado o ambiente, o professor aplica o modelo em sala de aula. Durante a aplicação, o conhecimento do aluno vai sendo desenvolvido nos três níveis (conhecer, compreender e aplicar), através dos diversos estímulos de aprendizagem que visam atingir diferentes perfis de alunos.

Assim, a partir da aplicação de abordagens de ensino focadas no aluno, potencializadas pela adoção de práticas de capacitação da indústria, espera-se que os alunos perpassem pelos diversos níveis de conhecimento até atingir o nível de aplicação, equivalente ao Profissional de Nível Iniciante, segundo o SWECOM (IEEE COMPUTER SOCIETY, 2014A). O SWECOM detalha as principais atividades esperadas para a unidade de Engenharia de Requisitos e o nível de aplicação dessas, conforme apresenta a **Tabela 9**.

Tabela 9 – Atividades Relacionadas à Unidade de Engenharia de Requisitos

Competência	**Atividades**	**Nível de Aplicação**
Elicitação de Requisitos de Software	1. Auxiliar no envolvimento das diferentes partes interessadas para determinar as necessidades e os requisitos. 2. Auxiliar na aplicação de diferentes métodos que sejam apropriados para obter requisitos.	Auxílio
Análise de Requisitos de Software	1. Auxiliar na análise de domínio.	Auxílio
Especificação de Requisitos de Software	1. Preparar a documentação de requisitos, incluindo as descrições de interfaces, requisitos funcionais e não funcionais.	Realização
Verificação e Validação de Requisitos de Software	1. Revisar especificações de requisitos para identificar erros e omissões.	Realização
Gerenciamento de Requisitos dos Processos e dos Produtos de Software	1. Auxiliar na aplicação de processos definidos para engenharia de requisitos.	Auxílio

Fonte: Adaptado do SWECOM (IEEE COMPUTER SOCIETY, 2014A).

Ao final da aplicação do modelo, o aluno deve estar apto a realizar essas 6 (seis) atividades de acordo com o nível de aplicação especificado. Para o nível de "Auxílio", o aluno deve ter a capacidade de auxiliar um profissional experiente na execução dessas atividades. Já para o nível "Realização", o aluno deve ter a capacidade de realizar essas atividades sem supervisão.

4.7 CONCLUSÃO

Este capítulo apresentou um modelo iterativo que possui como principal diferencial a integração das principais estratégias práticas adotadas para o ensino de Engenharia de Software e a adaptação de práticas de capacitação da indústria para o

contexto acadêmico. Todas essas abordagens e práticas podem ser classificadas como humanistas, pois são focadas no aluno.

Inicialmente, apresentaram-se os princípios que orientam o ensino prático de ES e, por consequência, fundamentam o modelo apresentado nesta pesquisa. Para situar essa pesquisa na área de ensino de ES, apresentaram-se as abordagens práticas que contribuíram diretamente tanto na fundamentação da tese quanto na definição da proposta do modelo.

Posteriormente, definiram-se as etapas do modelo apresentadas nesse capítulo, derivadas dos resultados obtidos na execução dos métodos de pesquisa. A partir da revisão da literatura e do *survey* com professores e alunos, identificaram-se as unidades de conhecimento e as abordagens práticas mais adotadas para o ensino de ES. Já a partir do levantamento com consultores, identificaram-se quais práticas de capacitação podiam ser adaptadas para o contexto acadêmico a fim de desenvolver competências técnicas de maneira mais adequada.

Esse capítulo também apresentou o sistema social que descreve a relação entre o professor e os alunos, bem como o papel desempenhado por cada um desses nas atividades de ensino-aprendizagem do modelo. De maneira similar, apresentou os requisitos e materiais de apoio necessários para o processo de instanciação e de desenvolvimento de competências nesse modelo.

O Capítulo 5 apresenta a avaliação desse modelo, tanto em relação à sua documentação e usabilidade quanto no que se refere ao seu apoio no desenvolvimento de competências técnicas em ES.

5 AVALIAÇÃO DO MODELO

5.1 INTRODUÇÃO

A fim de avaliar a especificação do modelo, aplicou-se em primeiro lugar o 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, inicialmente, de forma individual, onde os especialistas leram o modelo e responderam um questionário estruturado. Em seguida, realizou-se uma reunião de consenso, a fim de resolver discordâncias e emitir um parecer conclusivo da avaliação.

Posteriormente, aplicou-se o método experimento controlado a fim de avaliar a adoção do modelo pelos professores em sala de aula. Esta etapa consistiu na comparação entre duas abordagens de ensino: uma tradicional focada no professor e uma humanista focada no aluno. Assim, aplicaram-se essas abordagens e avaliaram-se a efetividade dessas em relação ao desenvolvimento de competências técnicas nos alunos.

Este capítulo apresenta o planejamento e a execução desses métodos de pesquisa, através do seu *design* e perfil dos participantes. Em seguida, descreve a execução, os resultados obtidos, bem como a análise, as discussões e as ameaças à validade desses resultados. Adicionalmente, analisa os resultados da avaliação desse modelo através da triangulação dos resultados do painel de especialistas, do *survey* e do experimento controlado. Por fim, destaca as principais lições aprendidas a partir desta avaliação.

5.2 PAINEL DE ESPECIALISTAS

5.2.1 *Design*

A avaliação da documentação do modelo de ensino foi realizada por meio de um painel de especialistas, um método empírico baseado na opinião de especialistas (BEECHAM et al., 2005). Os critérios para a seleção dos especialistas foram: ter experiência docente e profissional na área de ES; possuir titulação a nível de doutorado; ter publicações de pesquisas relacionadas ao ensino de ES.

As perguntas de pesquisa dessa etapa foram derivadas do mapeamento sistemático de Marques, Quispe e Ochoa (2014), que destaca que os professores considerem 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 do tipo *design* que buscam avaliar a efetividade da especificação do modelo conforme apresenta a **Tabela 10**Erro! Fonte de referência não encontrada..

Tabela 10 – Perguntas da Avaliação por Painel de Especialistas

Categoria	Critério	Pergunta
Documentação do Modelo	Adequação	*O texto da documentação está adequado ao tema abordado pelo modelo (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 modelo de ensino?*
	Consistência	*Os componentes estão consistentes com o foco do modelo de ensino?*
	Corretude	*A documentação descreve os componentes do modelo 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 modelo de ensino?*
Usabilidade do Modelo	Facilidade de Uso	*O modelo é fácil de usar?*
	Flexibilidade do Modelo	*O modelo permite flexibilidade no seu uso?*
	Adequação do Modelo	*O modelo é adequado para ser usado pela maioria dos professores?*
	Esforço Envolvido	*Qual o grau de esforço envolvido para um professor usar o modelo?*
	Habilidades Requeridas	*Qual o grau de habilidades requeridas para um professor usar o modelo?*
	Restrições de Uso	*Qual o grau de restrições para usar o modelo?*
Parecer Conclusivo	Utilidade do Modelo	*O quão útil você considera o modelo apresentado no apoio ao ensino de Engenharia de Software?*
	Recomendação do Modelo	*O quanto você recomendaria esse modelo para uso em sala de aula no ensino de Engenharia de Software?*
	Pontos Fracos do Modelo	*De maneira geral, quais são os pontos fracos do modelo avaliado?*
	Pontos Fortes do Modelo	*De maneira geral, quais são os pontos fortes do modelo avaliado?*
	Considerações Finais	*Quais são as recomendações gerais do(a) avaliador(a) para os proponentes do modelo?*

Fonte: Elaborado pelo autor.

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 subjetivas 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. Adicionalmente, realizaram-se 3 perguntas subjetivas no parecer final sobre os pontos fortes, pontos fracos do modelo e considerações finais. O questionário de avaliação, com todas essas perguntas, encontra-se disponível em *https://goo.gl/9JBvvV*.

O processo de avaliação ocorreu em duas etapas. Inicialmente, houve uma análise e avaliação individual do modelo (via questionário no *Google Forms*) no período de 13 de dezembro de 2016 a 17 de janeiro de 2017. Em seguida, foi feita uma reunião de consenso (assíncrona, via *Google Docs*) entre os especialistas, no período de 16 a 24 de fevereiro de 2017, onde estes justificaram as notas atribuídas na avaliação. Essa reunião objetivou resolver as discordâncias, possibilitando o desenvolvimento de um consenso final do painel de especialistas. Como resultado, foi emitido um parecer conclusivo da avaliação.

5.2.2 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 o professor Adriano Albuquerque[5] da Universidade de Fortaleza (UNIFOR), o professor Alberto França[6] da Universidade Federal Rural de e a professora Renata Moreira[7] da Universidade de Lavras (UFLA). Quanto à experiência, todos os especialistas já realizaram projetos práticos, discussão de casos práticos, dinâmicas e *workshops* no ensino de ES. Dois desses já adotaram PBL, jogos educativos e *mentoring* em sala de aula e apenas um aplicou g*amification* e *coaching*.

5.2.3 Resultados

A partir da reunião de consenso, conclui-se que o painel de especialistas considerou o modelo completo e consistente. Os professores também destacaram a flexibilidade e a utilidade do modelo para o ensino de ES, conforme **Tabela 11**.

[5] http://lattes.cnpq.br/2680368743615023
[6] http://lattes.cnpq.br/2939395207680085
[7] http://lattes.cnpq.br/7640822964644158

Tabela 11 – Resultados da Avaliação por Painel de Especialistas

Categoria	Critério	Parecer do Painel de Especialistas
Documentação do Modelo	Adequação	SIM
	Clareza	NÃO
	Completude	SIM
	Consistência	SIM
	Corretude	SIM
	Detalhamento	NÃO
Usabilidade do Modelo	Facilidade de Uso	Discordam
	Flexibilidade do Modelo	Concordam Totalmente
	Adequação do Modelo	Concordam Parcialmente
	Esforço Envolvido	Muito Esforço
	Habilidades Requeridas	Habilidades Avançadas
	Restrições de Uso	Restrições Moderadas
	Utilidade do Modelo	Muito Útil
	Recomendação do Modelo	Recomendariam Largamente

Fonte: Elaborado pelo autor.

De acordo com a análise do parecer final, os especialistas consideraram que o modelo possui “baixa usabilidade, falta de clareza e objetividade”. No entanto, destacaram que os pontos fortes do modelo são a sua “simplicidade e linguagem apropriada ao público de ES”. Desta forma, consideram que “o modelo se caracteriza como um material que o professor, que não é especialista em pedagogia, pode contar”.

Como sugestões de melhoria, os avaliadores destacaram que o modelo “precisava melhorar bastante a facilidade de uso e o seu conteúdo, pois muitos professores não terão paciência de ler e reler o modelo até poder utilizá-lo”. Por fim, consideraram que o mesmo “representa bem o ensino de ES nos dias atuais” e que o problema na sua adoção pode ser “a mudança de paradigma para aqueles professores que estão acostumados com o ensino tradicional”.

5.2.4 Análise e Discussões

Em relação à documentação do modelo, a fim de tornar o seu texto mais claro e objetivo para uso em sala de aula, os autores revisaram a documentação e diminuíram a

parte da fundamentação. Inicialmente, na versão 1.0, o modelo possuía 78 páginas. Com essas alterações, a versão 2.0 passou a ter 40 páginas. Adicionalmente, desenvolveu-se um sistema *web* do modelo, disponível em *http://carlosportela.com.br/lafoca/*, que sintetiza esse texto, a fim de torna-lo ainda mais objetivo, e disponibiliza o download da documentação completa.

Quanto à falta de usabilidade da documentação do modelo, o sistema permite o uso e instanciação desta. Em relação às representações gráficas, o sistema contém, na seção de Ajuda, exemplos ilustrados para melhor elucidar a instanciação de cada etapa do modelo.

Os avaliadores recomendaram o uso e exemplos a fim de melhorar o detalhamento do modelo. Assim, o sistema proposto permite a submissão de material de apoio (artigos, vídeos, dinâmicas, etc.) para cada etapa, bem como o cadastro de exemplos de problemas e perguntas relacionadas à unidade de conhecimento. Na avaliação, os especialistas consideraram que era necessário muito esforço para preparar o material de aula. Neste sentido, o sistema do modelo permite agregar o material de apoio disponibilizado pelos professores usuários.

Relacionado à disposição do professor para mudar seu paradigma de ensino, foi inserido uma subseção na Introdução do documento do modelo a fim de explicitar o Perfil do Professor que é público-alvo do modelo. Por fim, para determinar quais habilidades são requeridas pelos professores que são público-alvo do modelo, foi inserida uma subseção na Introdução do documento para explicitar os Pré-Requisitos de uso do modelo, como por exemplo, experiência na indústria.

5.2.5 Ameaças à Validade

Dentre as ameaças à validade desta avaliação tem-se o fato do painel ter envolvido apenas 3 especialistas, o que afetam à validade de conclusão. Neste sentido, destaca-se que essa amostra representa apenas 11% da população de professores participantes do *survey*. No entanto, destaca-se a média de experiência docente de 8 anos desses especialistas, bem como o conhecimento destes na aplicação das abordagens discutidas neste trabalho.

Adicionalmente, há uma ameaça relacionada ao fato dos especialistas apenas terem analisado a documentação do modelo, o que pode ter limitado seu entendimento,

levando a avaliações equivocadas. Nesse sentido, utilizou-se a escala *Likert* nas questões objetivas, permitindo-se a inserção de considerações sobre os valores escolhidos, a fim de complementar a avaliação dos especialistas. Por fim, realizou-se uma reunião de consenso a fim de discutir eventuais equívocos em relação ao entendimento do modelo proposto.

5.3 EXPERIMENTO CONTROLADO

5.3.1 *Design*

De acordo com Easterbrook et al. (2007), uma pré-condição para conduzir um experimento controlado é ter claro quais hipóteses serão observadas. Tradicionalmente, trabalha-se com hipóteses nula e alternativa que são duas declarações mutuamente exclusivas sobre uma população. A Hipótese Nula (H_0) afirma que um parâmetro da população é igual a um valor hipotético. Já a Hipótese Alternativa (H_1) afirma que um parâmetro da população é menor, maior ou diferente do valor hipotético na hipótese nula. A hipótese alternativa é aquela que se acredita que pode ser verdadeira ou que se espera provar ser verdadeira.

Essas hipóteses guiaram todas as etapas do experimento, incluindo quais variáveis foram mensuradas. Assim, buscando responder à Questão de Pesquisa (QP) apresentada na Seção 1.3, definiu-se a seguinte hipótese nula:

> **H_0:** *A abordagem humanista de ensino permite o desenvolvimento de competências técnicas em Engenharia de Software em nível de aplicação de maneira menos eficiente, ou igual, do que a abordagem tradicional de ensino.*

Na hipótese H_0, o tratamento é a "abordagem humanista de ensino" e o controle é a "abordagem tradicional de ensino". Nesse cenário, a hipótese alternativa a ser observada seria:

> **H_1:** *A abordagem humanista de ensino permite o desenvolvimento de competências técnicas em Engenharia de Software em nível de aplicação de maneira mais eficiente do que a abordagem tradicional de ensino.*

Essas hipóteses podem ser formalizadas a partir da seguinte representação:

> **H_0 =** abordagem humanista de ensino ≤ abordagem tradicional de ensino
>
> **H_1 =** abordagem humanista de ensino > abordagem tradicional de ensino

Na análise dessas duas hipóteses, foi observada a variável dependente competência técnica em Engenharia de Software, que foi medida usando como referência o processo de avaliação do SWECOM (IEEE COMPUTER SOCIETY, 2014A), para determinar se a variável independente abordagens de ensino possui efeito no resultado final. A variável abordagens de ensino foi instrumentalizada a partir das revisões sistemáticas de Malik e Zafar (2012), Marques, Quispe e Ochoa (2014) e Santos et al. (2014).

Da mesma forma, as variáveis independentes abordagem humanista de ensino e abordagem tradicional de ensino, classificadas a partir do trabalho de Mizukami (1986), foram manipuladas no experimento a fim de avaliar a sua influência na variável dependente. Sendo assim, indiretamente, buscou-se responder a seguinte pergunta do tipo causalidade-comparação: "A abordagem humanista é mais efetiva para o desenvolvimento de competências técnicas em ES do que a abordagem tradicional?".

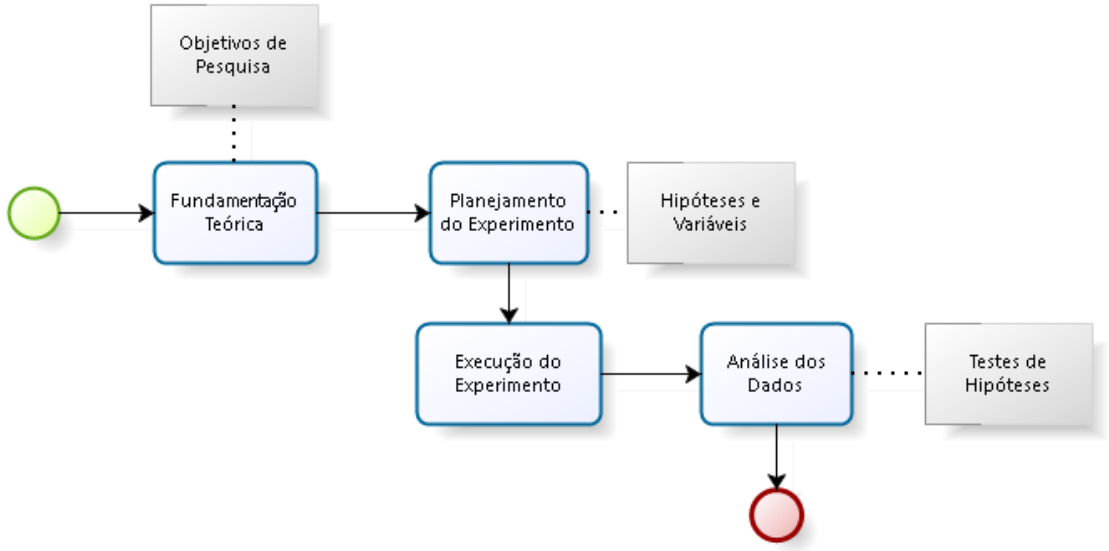
Este experimento controlado foi aplicado em um curso de graduação em Sistemas de Informação da Universidade Federal do Pará (UFPA), Campus de Cametá, durante o primeiro semestre de 2017. As disciplinas-alvo foram "Gerência de Projetos de Software", cuja unidade de conhecimento foco do experimento foi Gerenciamento de Projetos, e "Análise e Projeto de Sistemas I", cuja unidade-foco foi Engenharia de Requisitos. O protocolo de pesquisa definido para esse experimento seguiu os *guidelines* de Jedlitschka, Ciolkowski e Pfahl (2007) e encontra-se no Apêndice F deste trabalho.

Quanto ao *design* de coleta de dados, esse experimento se classifica como 3x3, no qual três grupos (amostras da população de estudo) recebem três tratamentos distintos e a coleta dos dados ocorre em dois momentos distintos no tempo (início e fim da disciplina) (KITCHENHAM e PFLEEGER, 2008). Os participantes foram selecionados a partir de uma análise do conhecimento prévio desses, onde procurou-se equilibrar o nível de competência entre os grupos. Assim, definiu-se um grupo de controle e dois grupos experimentais (tratamento 1 e 2).

A fim de planejar e executar o experimento controlado desta pesquisa foram seguidas 4 (quatro) etapas, conforme ilustra a **Figura 19**. Inicialmente, na *Etapa I – Fundamentação Teórica* foram identificados os problemas, os objetivos e o contexto de pesquisa a fim de localizar as informações necessárias para as próximas etapas do experimento controlado. Em seguida, foram identificados os estudos relacionados às hipóteses e variáveis envolvidas a fim de fundamentar o experimento. Nessa etapa,

identificaram-se as abordagens de ensino de Gnatz et al. (2003) e Garg e Varma (2008), que propõem a adaptação de práticas da indústria para o contexto acadêmico, a análise de abordagens focadas no aluno feita por Prikladnicki et al. (2009), além do modelo pedagógico de Gary et al. (2013). Na *Etapa II – Planejamento do Experimento* definiram-se os objetivos do experimento, a amostra de estudo, a operacionalização das variáveis, o *design* e os procedimentos de análise de dados, nesse caso, a análise de variância ANOVA (EASTERBROOK et al., 2007).

Figura 19 – Etapas da Realização do Experimento Controlado



Fonte: Adaptado de Jedlitschka, Ciolkowski e Pfahl (2007).

Na *Etapa III – Execução do Experimento*, realizou-se o treinamento dos professores no modelo (abordagem humanista de ensino) em Abril de 2017. Em seguida, foram aplicados questionários pré e pós-aplicação das unidades de conhecimento de ES a fim de aferir a aprendizagem dos alunos. Esses questionários foram aplicados como forma de averiguar o conhecimento prévio dos alunos e formar os grupos do experimento de forma nivelada. Por fim, na *Etapa IV – Análise dos Dados* buscou-se identificar o grau de aprendizagem dos alunos nas duas abordagens de ensino (tradicional e humanista), de acordo com as respostas fornecidas nos questionários e entrevistas dos professores e alunos participantes do experimento.

Foram utilizados como instrumentos para aplicação do experimento: questionários auto-administrados, entrevistas estruturadas, dicionários de dados e mapas conceituais. Esses instrumentos foram aplicados tanto no grupo de controle, que seguiu uma abordagem tradicional baseada na abordagem adotada pelo professor na disciplina,

quanto nos grupos do experimento, que seguiram abordagens focadas no aluno e práticas de capacitação da indústria.

Os questionários utilizaram as mesmas questões de pesquisa do *survey* (ver Subseção 3.5.1) relacionadas à aprendizagem de tópicos, a fim de, posteriormente, realizar a triangulação dos resultados. No entanto, os questionários do experimento abrangeram apenas os tópicos das unidades foco do experimento: Gerenciamento de Projetos e Engenharia de Requisitos. Esses questionários foram disponibilizados no *Google Forms* no início e após a conclusão da disciplina.

As entrevistas estruturadas foram baseadas no procedimento de avaliação de competências do SWECOM (IEEE COMPUTER SOCIETY, 2014A). Para cada competência, realizam-se perguntas relacionadas às habilidades técnicas do aluno/profissional. Essas entrevistas foram conduzidas por um consultor que possui mais de 10 anos de atuação na área de capacitação profissional em ES.

Adicionalmente, os dicionários de dados e mapas conceituais foram utilizados como instrumentos de avaliação qualitativa da aprendizagem dos alunos. Os dicionários de dados criados pelos alunos permitem analisar os conceitos absorvidos por esses em relação aos tópicos de ES. Já os mapas conceituais permitem analisar como os alunos relacionam esses conceitos entre si.

Por fim, solicitou-se aos professores que aplicaram o modelo que avaliassem a sua documentação e usabilidade, seguindo os mesmos critérios adotados no painel de especialistas. Assim, pôde-se triangular os resultados obtidos nessas duas fases de avaliação do modelo, conforme destaca a Seção 5.4.

A. Fase 1 – Aplicação do Modelo na Unidade de Gerenciamento de Projetos

A primeira fase do experimento, na disciplina de "Gerência de Projetos de Software", foi realizada durante o período de 26 de Abril a 13 de Maio de 2017 no Laboratório de Informática que fica nas dependências da UFPA Cametá, no período das 14h00 às 18h00. A turma foi dividida em 3 (três) grupos. Esses grupos foram divididos de acordo com os resultados obtidos no questionário de percepção de conhecimento dos tópicos da unidade de Gerenciamento de Projetos. Assim, gerou-se um *ranking*, onde o 1º colocado foi alocado para o Grupo A, o 2º para o Grupo B e o 3º para o Grupo C. O 4º foi alocado para o Grupo C, o 5º para o Grupo B e o 6º para o Grupo A e assim

sucessivamente, objetivando o balanceamento entre as equipes. Cada grupo foi composto por 4 (quatro) alunos.

O Grupo A seguiu a abordagem tradicional, sendo o grupo de Controle, tendo aulas sobre a unidade de Gerenciamento de Projetos no período das 14h00 às 16h00. Já o Grupo B seguiu a abordagem humanista através das práticas focadas no aluno do modelo de ensino, sendo o grupo do Tratamento 1. O Grupo C também seguiu a abordagem humanista, mas além das práticas focadas no aluno adotou as práticas de capacitação da indústria, sendo o grupo do Tratamento 2. Dessa forma, durante a pesquisa, os alunos do Grupo C foram auxiliados por alunos veteranos que realizaram *mentoring* e pelo professor que realizou *coaching*. Ambos grupos B e C tiveram aula no período das 16h00 às 18h00. Sendo assim, o tempo de cada seção do experimento não ultrapassou 2 horas.

A fim de controlar o experimento e diminuir os vieses, o mesmo professor que ministrou as aulas para o grupo de controle o fez para os grupos de tratamento. Assim, esse professor realizou um treinamento referente ao uso da abordagem de ensino iterativa. Ao final do experimento, o professor avaliou o modelo seguindo o questionário do painel de especialistas (ver Tabela 10).

Antes de iniciar o experimento, os alunos preencheram um questionário sobre o conhecimento prévio dos tópicos de ES, foram entrevistados pelo especialista em relação às suas competências e criaram dicionários de dados e mapas conceituais sobre os tópicos da unidade abordada. Em seguida, o professor ministrou as aulas tradicionais para o Grupo A e, posteriormente, seguiu as etapas do modelo para os grupos B e C.

Ao final da disciplina, os alunos preencheram novamente o questionário sobre o grau de aprendizado obtido para os tópicos de ES, realizaram a entrevista com especialista a fim de identificar as competências obtidas, e criaram novos dicionários de dados e mapas conceituais. O objetivo foi comparar as respostas fornecidas pré e pós-disciplina, ou seja, a aprendizagem adquirida a partir do seguimento do controle e tratamentos.

B. Fase 2 – Aplicação do Modelo na Unidade de Engenharia de Requisitos

Foi realizado uma segunda fase do experimento, que seguiu o mesmo protocolo e passos que o anteriormente relatado. Esse experimento envolveu a disciplina de "Análise e Projeto de Sistemas I" durante o período de 27 de Junho a 14 de Julho de 2017 no mesmo Laboratório de Informática da UFPA Cametá, no período das 08h00 às 12h00. O Grupo A (Controle) teve aulas sobre a unidade de Engenharia de Requisitos no período

das 08h00 às 10h00. Já o Grupo B (Tratamento 1) e o Grupo C (Tratamento 2) tiveram aula no período das 10h00 às 12h00.

Essa segunda fase foi realizada buscando reforçar os resultados obtidos no primeiro e aumentar a população do estudo. Assim, somando as 2 (duas) fases participaram da avaliação do modelo 2 professores com titulação de mestre e 30 alunos. Essa amostra da população participante representa uma parcela de 42,85% da amostra entrevistada no *survey* (70 participantes). As ameaças dessa baixa amostragem são discutidas na Subseção 5.3.6.

5.3.2 Participantes

Como público-alvo do experimento controlado, definiu-se:

I. Professores que estão lecionando a disciplina da área de ES;

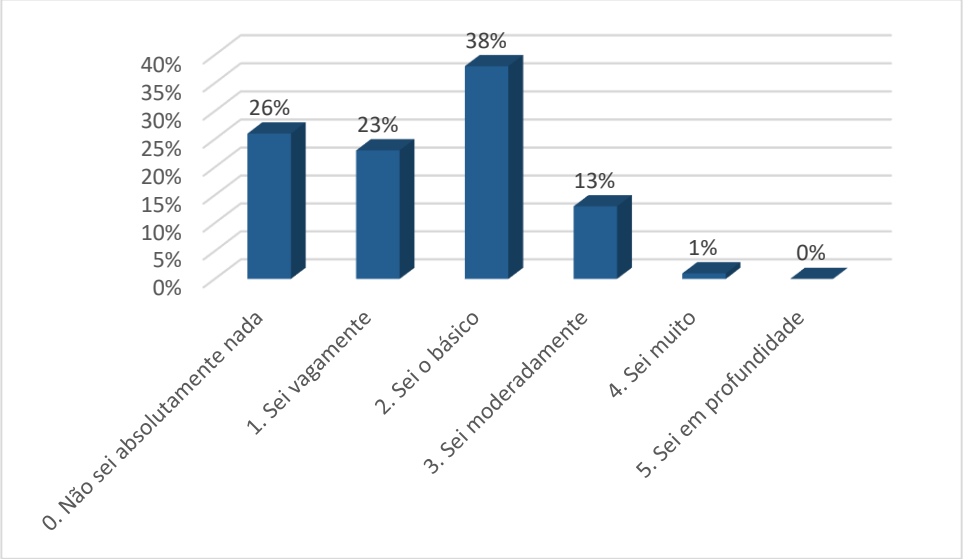
II. Aluno(a)s que estão cursando a disciplina da área de ES.

Foi apresentado a esse público-alvo do experimento um Termo de Consentimento Livre e Esclarecido (TCLE) a fim de obter o comprometimento com a pesquisa. Este termo explica o propósito e as etapas do experimento controlado. Somente os alunos que assinaram este termo participaram do experimento. É importante destacar que estes estudantes já tinham uma experiência acadêmica prévia, pois já haviam cursado disciplinas da área de Engenharia de Software.

Assim, os participantes do experimento foram alunos do curso de graduação em Sistemas de Informação da Universidade Federal do Pará (UFPA), Campus de Cametá, do primeiro semestre de 2017. Na Fase 1 do experimento, participaram 12 alunos da disciplina de “Gerência de Projetos de Software”. Já na Fase 2, participaram 18 alunos da disciplina de “Análise e Projeto de Sistemas I”.

É importante destacar que essas duas turmas de alunos possuem perfis bastante distintos. Os alunos da Fase 1 estão cursando o 6º semestre do curso, onde a maioria possui vínculo com projetos de pesquisas da universidade. Isso faz com que estes possuam certa maturidade para lidar com responsabilidade e realizar trabalhos em equipe. Essa característica contribuiu para os resultados obtidos durante o experimento, principalmente devido a unidade dessa fase ter sido Gerenciamento de Projeto. No entanto, esses alunos não possuíam um grande conhecimento prévio dessa unidade, conforme mostram os percentuais do **Gráfico 13**.

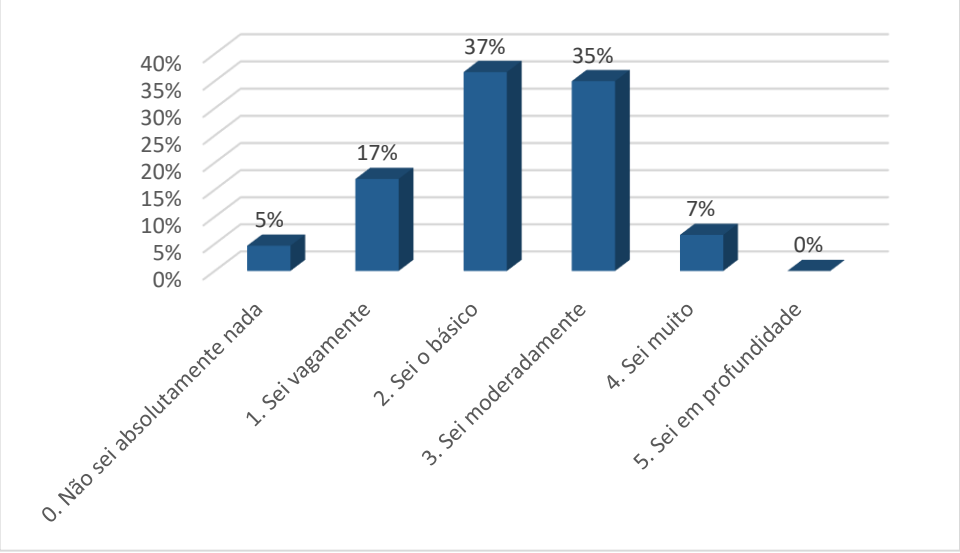
Gráfico 13 – Conhecimento Prévio dos Alunos da Fase 1 do Experimento



Fonte: Elaborado pelo autor.

Já os alunos da Fase 2, que estão cursando o 5º do curso, são mais novos que os da Fase 1, onde a maioria não possui experiência extraclasse. Isso faz com que a maioria destes não demonstre comprometimento com as atividades do curso. Essa característica impactou nos resultados obtidos, principalmente devido à falta de responsabilidade com as atividades do experimento. Apesar disso, esses alunos apontaram que possuem um conhecimento prévio mediano na unidade de Engenharia de Requisitos, se comparado com os alunos da Fase 1, conforme mostram os percentuais do **Gráfico 14**.

Gráfico 14 – Conhecimento Prévio dos Alunos da Fase 2 do Experimento



Fonte: Elaborado pelo autor.

Apesar das diferenças entre o conhecimento técnico das duas turmas, destaca-se que o objetivo do experimento foi analisar a aplicação do modelo em uma unidade técnica (Engenharia de Requisitos) e uma não-técnica (Gerenciamento de Projetos). Desta forma, essas unidades apresentam complexidade diferente, não sendo possível correlacionar diretamente o conhecimento prévio com os resultados obtidos em cada fase do experimento.

5.3.3 Execução do Experimento

Inicialmente, os professores realizaram a preparação da disciplina para aplicação do modelo, antes do início das aulas, a fim de definir as unidades de conhecimento abordadas nesse experimento e, consequentemente, selecionar as competências técnicas a serem desenvolvidas pelos alunos. Nessa etapa de preparação, os professores selecionaram o material instrucional, como os artigos, videoaulas, dinâmicas e jogos e planejaram o projeto prático a ser desenvolvido.

Adicionalmente, os professores identificaram alunos do curso que já tiveram experiência como gerente de projetos e analista de requisitos a fim de realizar as atividades de *mentoring*. Por fim, os professores discutiram entre si como seriam as intervenções relacionadas ao *coaching*. Após realizar essas atividades, ambos os professores cadastraram a disciplina-alvo da aplicação do modelo no sistema.

Na Fase 1, o professor selecionou como foco do projeto prático da disciplina a criação de um Plano de Projeto do desenvolvimento de um Portal da Prefeitura de Oeiras. Assim, na primeira aula, relativa à etapa de Iniciação, apresentou a seguinte problemática: "*Como gerenciar um projeto de software a fim de que ele obtenha sucesso?*".

Nessa aula, os alunos discutiram juntamente com o professor ideias de como solucionar essas problemáticas. Assim, o professor pode excitar o papel de *coach* através da sugestão e explicação de técnicas específicas, como entrevistas e *brainstorm*, por exemplo. Posteriormente, na segunda aula, na etapa de Preparação, o professor indicou videoaulas sobre os principais conceitos de Gerência de Projetos e a leitura de artigos técnicos com estudos de caso da aplicação de Scrum em empresas de desenvolvimento de software.

Na terceira aula, na etapa de Discussão, o aluno convidado, com experiência como gerente de projetos, ministrou um *workshop* sobre o *framework* ágil Scrum, tirando

dúvidas e fomentando discussões entre os demais alunos. Na quarta aula, na etapa de Prática, o professor realizou a dinâmica denominada “Fábrica de Aviões”, que permite aos alunos vivenciar um processo de desenvolvimento baseado em metodologias ágeis.

Finalmente, na quinta aula, deu-se início à realização do projeto prático, na etapa de Contextualização, onde os alunos puderam entrevistar o cliente que, neste caso, era o *Product Owner* do projeto do Portal da Prefeitura de Oeiras. No total, os alunos tiveram 6 aulas (3 semanas) para realizar o projeto prático. Assim, aplicando o *framework* ágil Scrum para gestão e planejamento do projeto, realizaram a estimativa de esforço e tempo através do *Planning Poker*. Ao final, nas duas últimas aulas, décima primeira e décima segunda, os alunos entregaram ao cliente os documentos de Visão do Produto, *Product Backlog*, Plano de Comunicação e o Cronograma do Projeto.

No total, a Fase 1 do experimento durou 6 semanas e o resultado obtido foi satisfatório, pois o cliente optou por adotar os Planos de Projetos elaborados na disciplina para o projeto real do Portal da Prefeitura de Oeiras.

Já na Fase 2, o professor selecionou como foco do projeto prático da disciplina a elicitação dos requisitos de um aplicativo *mobile* para cursinhos pré-vestibulares. Assim, na aula relativa à etapa de Iniciação, apresentaram-se as seguintes problemáticas:

1. *Como identificar os stakeholders do projeto?*
2. *Como coletar as necessidades do cliente?*
3. *Como registrar os requisitos do projeto*?

Ainda nessa primeira aula, os alunos expuseram suas ideias de solução e as discutiram com o professor, que exerceu o papel de *coach* sugerindo técnicas específicas, como *storyboards* e prototipagem, por exemplo. Na segunda aula, na etapa de Preparação, o professor indicou videoaulas sobre a Elicitação de Requisitos e a leitura de artigos técnicos sobre a diferença entre Requisitos Funcionais e Não-Funcionais.

Na terceira aula, na etapa de Discussão, o aluno convidado, com experiência de mercado como analista de requisitos, ministrou um *workshop* sobre *Como Elicitar Requisitos na Indústria*, gerando discussões na turma. Na etapa de Prática, durante a quarta aula, o professor realizou a dinâmica denominada “Desenhando Diagramas”, que que consiste em elaborar diagramas, semelhantes à uma página *web*, a partir da descrição de um aluno-cliente, o único que visualizou o diagrama finalizado. Assim, os demais

tentam, através de técnicas diversas como entrevista e prototipagem, representar o diagrama descrito pelo cliente.

Na quinta aula, iniciou-se o projeto prático, na etapa de Contextualização, onde os alunos puderam entrevistar o cliente que, neste caso, era o idealizador do projeto do aplicativo para gerenciamento de cursinhos pré-vestibulares. No total, os alunos tiveram 4 aulas (2 semanas) para realizar o projeto prático. Neste sentido, puderam aplicar técnicas de elicitação de requisitos como a modelagem de casos de uso, diagramas de classes e modelo entidade-relacionamento. Adicionalmente, a partir da sugestão do profissional convidado, elaboraram *storyboards* e protótipos de telas, que objetivam ajudar os clientes a compreender melhor os cenários e funcionalidades necessárias para o aplicativo. Por fim, na nona e décima aula, os alunos entregaram ao cliente um documento de Lista de Requisitos.

No total, a Fase 1 do experimento durou 4 semanas, sendo o resultado do mesmo elogiado pelo idealizador do projeto, que destacou o uso de *storyboards* e protótipos. Ao final da disciplina, esse idealizador demonstrou interesse em utilizar os *storyboards* e protótipos gerados na documentação de requisitos a ser apresentada ao cliente do projeto.

5.3.4 Análise Quantitativa

Realizou-se a análise dos dados coletados pré e pós-experimento utilizando análise estatística. Assim, para avaliar a diferença entre os resultados de um mesmo instrumento pré e pós-experimento, utilizou-se o Teste T de *Student*, recomendado quando a estatística de teste segue uma distribuição normal, mas a variância da população é desconhecida. Nesse caso, é usada a variância amostral e, com esse ajuste, a estatística de teste passa a seguir uma distribuição T de *Student*.

Na Fase 1 do Experimento, conforme mostra a **Tabela 12**, o valor de P para a diferença dos dicionários de dados, mapas conceituais pré e pós-experimento foi maior que 0,05, ou seja, não houve diferença significativa. Quanto às entrevistas, o valor de P em todos os grupos foi menor que 0,05. Assim, pode-se afirmar que houve desenvolvimento de competências, independente da abordagem (Controle, Tratamento 1 e Tratamento 2).

Tabela 12 – Análise dos Resultados do Experimento Controlado 1

Instrumento	**Grupo A (Controle)**	**Grupo B (Tratamento 1)**	**Grupo C (Tratamento 2)**
Dicionário de Dados	P = 0,26603	P = 0,15407	P = 0,17528
Mapa Conceitual	P = 0,39100	P = 0,82400	P = 0,10272
Entrevista	P = 0,01384	P = 0,00581	P = 0,00567

Fonte: Elaborado pelo autor.

Na Fase 2 do Experimento, conforme mostra a **Tabela 13**, o valor de P para a diferença dos dicionários de dados, mapas conceituais foi menor que 0,05. Assim, pode-se afirmar que houve diferença significativa de ganho de aprendizagem de conceitos, independente da abordagem (Controle, Tratamento 1 e Tratamento2). Quanto às entrevistas, apenas o Grupo C apresentou uma diferença significativa, ou seja, houve impacto do tratamento 2 (práticas de capacitação da indústria) no desenvolvimento de competências.

Tabela 13 – Análise dos Resultados do Experimento Controlado 2

Instrumento	**Grupo A (Controle)**	**Grupo B (Tratamento 1)**	**Grupo C (Tratamento 2)**
Dicionário de Dados	P = 0,00037	P = 0,00017	P = 0,00399
Mapa Conceitual	P = 0,01172	P = 0,01677	P = 0,03280
Entrevista	P = 0,07775	P = 0,27217	P = 0,00601

Fonte: Elaborado pelo autor.

A fim de verificar se houve diferença significativa no desenvolvimento de competências técnicas de ES entre as diferentes abordagens, utilizou-se a Análise de Variância (ANOVA). Quando os resultados da ANOVA levam à rejeição da hipótese nula (abordagem humanista de ensino ≤ abordagem tradicional de ensino), que representa a afirmação de que todas as médias são iguais ou que o tratamento é menor que o controle, temos evidências de que as médias entre os níveis diferem significativamente.

A partir dessa análise, observou-se na Fase 1 do Experimento que houve diferença significativa entre os 3 grupos, sendo P = 0,00153. Adicionalmente, realizou-se análise de Teste T entre as diferentes abordagens. Os resultados obtidos ajudam a reforçar a hipótese alternativa (abordagem humanista de ensino > abordagem tradicional de ensino), pois a diferença entre o Tratamento 1 e Controle foi P = 0,01016 e entre o Tratamento 2

e Controle foi P = 0,00606. Não houve diferença significativa entre o Tratamento 1 e 2, sendo o valor de P = 0,18905.

No entanto, esses resultados não se repetiram na Fase 2 do Experimento, sendo o valor de P da ANOVA = 0,63130. Considerando que, nessa turma, o Grupo C apresentou a melhor diferença significativa (P = 0,00601) no Teste T de *Student*, pode-se inferir que o resultado em comum entre as duas fases do experimento foi o ganho significativo no desenvolvimento de competências técnicas quando se adota práticas de capacitação da indústria. A análise detalhada desses resultados encontra-se no Apêndice F deste trabalho.

5.3.5 Análise Qualitativa

De acordo com os professores que adotaram o modelo, a área de ES é "muito teórica e detalhada" e o modelo permite que os alunos "vivenciem uma prática dos assuntos", o que torna as aulas bem "mais interessantes e motivadoras". Além disso, destacaram que a possibilidade de aulas diferenciadas com vídeos, artigos, *workshops*, permite uma "maior interação entre os alunos" e torna a discussão e apresentação de dúvidas bem "mais presente e interessante".

Apesar dos dois professores concordarem que o modelo é fácil de usar, apontaram que a documentação poderia ser um pouco mais resumida para ser mais objetiva. Segundo um desses professores, "o texto é bastante detalhado", sugerindo que o mesmo fosse um pouco mais objetivo poderia facilitar para o entendimento. Neste sentindo, realizou-se uma revisão no texto, juntamente com os dois professores participantes do experimento, a fim de sintetizá-lo. Essas alterações implicaram na versão atual da documentação do modelo, a versão 3.0 que possui 34 páginas.

Por fim, os professores destacaram que os usuários do modelo precisam estar dispostos a inovar na didática para identificar e selecionar artigos, vídeos, profissionais e dinâmicas. Além disso, precisam conhecer e estar ciente das práticas de *coaching* e *mentoring* para fazer uso correto do modelo. Quanto à disposição para inovar na didática, o modelo definir um perfil específico como público-alvo do modelo, ou seja, professores humanistas que atuem como facilitadores do processo de ensino-aprendizagem. Relacionado à seleção de material de aula, o sistema do modelo disponibiliza uma base de artigos, vídeo e dinâmicas que podem ser instanciados em sala de aula. Já as práticas de *coaching* e *mentoring* foram adaptadas para o contexto acadêmico e suas etapas principais são descritas na documentação do modelo.

Os alunos participantes da fase do experimento relativa à unidade de Engenharia de Requisitos destacaram que a realização das atividades práticas, como dinâmicas de entrevistas, *workshop* sobre elicitação de requisitos através da modelagem de casos de uso, elaboração de diagramas de classes e modelo entidade-relacionamento e a sugestão do profissional convidado para que criassem *storyboards* e protótipos de telas, permitiu um melhor entendimento dos conceitos abordados na disciplina. Já na unidade de Gerenciamento de Projetos, os alunos destacaram que o projeto prático do modelo permitiu conhecer e aplicar o *framework* ágil Scrum para gestão e planejamento do projeto e a técnica de estimativa de esforço e tempo *Planning Poker*. Além disso, os alunos relatam que puderam compreender e elaborar os documentos de Visão do Produto, *Product Backlog*, Plano de Comunicação e o Cronograma do Projeto.

No entanto, pôde-se observar a partir da realização do experimento controlado que a falta de motivação e de comprometimento dos alunos impactaram diretamente nos resultados da instanciação do modelo. De acordo com relatos dos alunos da Fase 1 do Experimento, as principais dificuldades foram “o gerenciamento do tempo disponível para o desenvolvimento das atividades” e o “não cumprimento dos prazos”. Já na Fase 2 do Experimento, a principal dificuldade relatada por 67% dos alunos foi a “falta de compromisso por parte da equipe”, como “ausência em reuniões de trabalho” e o “não comprometimento com as tarefas a serem realizadas”.

5.3.6 Ameaças à Validade

Para o experimento, o fato dos dados analisados apontarem que uma abordagem humanista focada no aluno possa ser mais efetiva que uma abordagem tradicional focada no professor não significa que a hipótese alternativa seja verdadeira, mas apenas que não existe evidência suficiente para rejeitá-la. Nesse experimento, o valor do tratamento foi significativo apenas na Fase 1 do Experimento, sendo necessárias algumas observações sobre a sua relevância na Fase 2 do Experimento.

Neste sentido, avaliou-se a validade de construção convergente que avalia o grau em que diferentes questões, que são destinadas a medir o mesmo conceito, obtêm resultados semelhantes. Nesse caso, o objetivo era medir o conceito “grau de aprendizagem” de três grupos distintos.

Outra ameaça relacionada à validade de construção é a possibilidade de um experimento controlado não ser adequado para correlacionar a adoção de abordagens de

ensino com o desenvolvimento de competências técnicas, pois entrevistas estruturadas retornam como resposta um tipo de percepção pessoal do entrevistado. Nesse caso, acredita-se que métodos alternativos de pesquisa empírica possam ser usados para complementar essas conclusões e proporcionar um entendimento mais profundo da influência do uso de um modelo iterativo de ensino no desenvolvimento de competências técnicas em ES, como, por exemplo, um Estudo de Caso.

Relacionada à validade interna do experimento, as ameaças estão relacionadas à seleção das amostras e das medidas adotadas para representar as variáveis sob observação. A seleção da variável abordagens de ensino foi feita com base em revisões sistemáticas na área de ensino de ES (MALIK e ZAFAR, 2012) (MARQUES, QUISPE e OCHOA, 2014) (SANTOS et al., 2014). Essa variável foi operacionalizada por meio dos conceitos de abordagem tradicional e abordagem humanista, conforme classificação de Mizukami (1986). Já a variável competência técnica em Engenharia de Software foi definida a partir do modelo SWECOM (IEEE COMPUTER SOCIETY, 2014A). Assim, essa variável foi operacionalizada por meio da avaliação de competência desse modelo, instrumentalizada através de um formulário e entrevista estruturada.

Em relação à validade de conclusão, ou seja, as amostras envolvidas na avaliação do modelo, participaram 2 professores e 30 alunos. Isso representa apenas 8% da população de professores e 64% dos alunos participantes do *survey*.

Ainda relacionado à validade de conclusão, utilizou-se análise estatística através do Teste T de *Student* para analisar a diferença entre as competências dos alunos pré e pós-experimento. Adicionalmente, utilizou-se a Análise de Variância (ANOVA) a fim de verificar se houve diferença significativa no desenvolvimento de competências técnicas de ES entre as diferentes abordagens.

Destaca-se que a baixa amostra da população do experimento controlado tende a reduzir a validade externa, pois os resultados obtidos sobre esses alunos e professores podem não se aplicar a outras instituições de ensino.

Destaca-se, no entanto, que os participantes do experimento representam a população ideal para testar o uso do modelo proposto: professores sem experiência de mercado e alunos sem afinidade com a área de ES. Adicionalmente, o tempo de realização do experimento foi controlado pelos pesquisadores, durando em média 2 (duas) semanas.

No caso da adoção do modelo fora de um ambiente controlado, o professor poderá instanciar o modelo conforme o tempo que julgar necessário.

Por fim, destaca-se que o experimento foi realizado em um ambiente real de sala de aula, no decorrer de diferentes disciplinas da área de ES: Gerência de Projetos de Software e Análise e Projeto de Sistemas I. Os materiais de apoio e os instrumentos utilizados estão disponíveis no sistema *web* do modelo, sendo possível acessá-los em *http://carlosportela.com.br/lafoca/*.

5.4 TRIANGULAÇÃO DOS RESULTADOS

De acordo com Easterbrook et al. (2007), realizar experimentos baseados em hipóteses possui vantagens e desvantagens. Uma vantagem é que a análise de dados baseada em hipóteses tende a reduzir os problemas de generalização. Por outro lado, uma desvantagem é que se decide antecipadamente quais variáveis serão ignoradas, como a experiência do professor na abordagem e a motivação do aluno, por exemplo. Essas podem vir a ser importantes fatores fora do ambiente do experimento.

Para restringir essas desvantagens, adotou-se no desenho de pesquisa desta tese uma abordagem multi-métodos (HESSE-BIBER, 2010), utilizando-se as mesmas questões de pesquisa do *survey* e do painel de especialistas no experimento controlado. Assim, pôde-se triangular os resultados obtidos em cada experimento a fim de reforçar as conclusões e completude do estudo. Isto traz, por consequência, maior credibilidade aos resultados desta pesquisa.

Conforme destacado na Subseção 5.3.4, os resultados obtidos na Fase 1 do Experimento demonstram que o modelo de ensino foi mais eficiente que a abordagem tradicional de ensino no que diz respeito ao desenvolvimento de competências técnicas em ES. No entanto, esses resultados não se repetiram na Fase 2 do Experimento. A fim de melhor entender esses resultados, reforçar as conclusões e a completude deste estudo, realizou-se a triangulação dos dados do *survey* e painel de especialistas com os dados coletados no experimento controlado.

Quanto à avaliação da documentação e usabilidade do modelo, a **Tabela 14** compara a avaliação realizada pelo painel de especialistas com a realizada pelos professores após o uso desse modelo (ao final do experimento).

Tabela 14 – Triangulação de Dados entre Painel de Especialistas e Experimento

Categoria	**Critério**	**Painel de Especialistas**	**Professores do Experimento**
Documentação do Modelo	Adequação	SIM	SIM
	Clareza	NÃO	SIM
	Completude	SIM	SIM
	Consistência	SIM	SIM
	Corretude	SIM	SIM
	Detalhamento	NÃO	SIM
Usabilidade do Modelo	Facilidade de Uso	Discordam	Concordam
	Flexibilidade do Modelo	Concordam Totalmente	Concordam Totalmente
	Adequação do Modelo	Concordam Parcialmente	Concordam Parcialmente
	Esforço Envolvido	Muito Esforço	Esforço Habitual
	Habilidades Requeridas	Habilidades Avançadas	Poucas Habilidades
	Restrições de Uso	Restrições Moderadas	Não Conseguem Avaliar
	Utilidade do Modelo	Muito Útil	Muito Útil
	Recomendação do Modelo	Recomendariam Largamente	Recomendariam Fortemente
Parecer Conclusivo	Pontos Fracos do Modelo	Baixa usabilidade, falta de clareza e objetividade.	O professor precisa estar disposto a inovar na didática e estar bem ciente dos termos como *coaching* e *mentoring* para fazer uso correto do modelo.
	Pontos Fortes do Modelo	Simplicidade e linguagem apropriada ao público de ES. Assim, o modelo caracteriza-se como um material que o professor, que não é especialista em pedagogia, pode contar.	O modelo permite que os alunos vivenciem uma prática dos assuntos, o que torna as aulas bem mais interessantes e motivadoras. Além disso, a possibilidade de aulas diferenciadas com vídeos, artigos, workshops, permite uma maior interação entre os alunos e torna a aula bem mais interessante.
	Considerações Finais	Precisa melhorar bastante a facilidade de uso do modelo e o conteúdo, pois muitos professores não terão paciência de ler e reler o modelo até poder utilizá-lo.	O texto está bem claro, mas o documento poderia ser um pouco mais resumido para ser mais objetivo. No geral, o texto é bastante detalhado, por isso acho que se fosse

Categoria	Critério	Painel de Especialistas	Professores do Experimento
		Mas representa bem o ensino de ES nos dias atuais. O problema pode ser a mudança de paradigma para aqueles professores que estão acostumados com o ensino tradicional.	um pouco mais objetivo poderia facilitar o seu entendimento.

Fonte: Elaborado pelo autor.

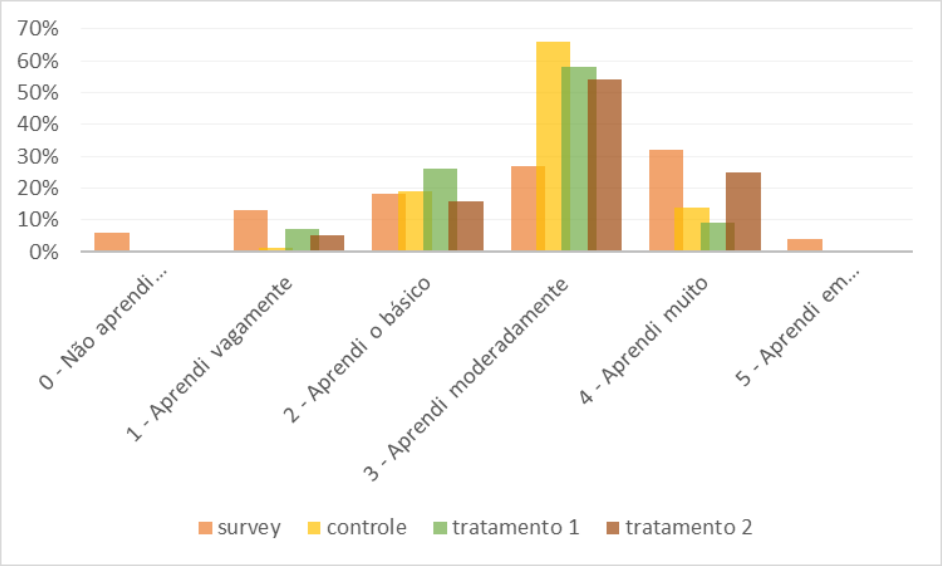
A partir da análise conjunta desses dados, pode-se afirmar que o modelo é adequado para o ensino de ES e completo, pois apresenta os componentes necessários e consistentes com o seu foco. Adicionalmente, a documentação está descrita de maneira correta, pois não se encontraram erros significativos de sintaxe ou semântica. No entanto, o modelo não foi considerado claro e objetivo, pois os avaliadores consideraram o nível de detalhamento bastante alto. Nesse sentido, ressalta-se que após a realização do painel de especialistas, foram realizadas alterações no modelo a fim de atender às recomendações do parecer final. Da mesma forma, após a realização do experimento controlado, foram realizadas novas alterações no modelo a fim de atender as recomendações dos professores que fizeram uso dele. Dessa forma, o modelo encontra-se atualmente na versão 3.0, disponível em *http://carlosportela.com.br/lafoca/*.

Quanto à usabilidade, os professores participantes do experimento concordaram que o modelo é fácil de usar, flexível, requerendo um esforço habitual e poucas habilidades dos professores para adotá-lo. Acredita-se que essa melhoria na usabilidade se deve ao sistema *web* do modelo, resultado das recomendações de melhoria do painel de especialistas.

Os professores que avaliaram o modelo concordam parcialmente que esse possa ser adequado para a maioria dos professores e possui restrições moderadas para sua adoção. Dessa forma, inseriu-se no modelo uma seção que define o perfil de professor que melhor se adequa como público-alvo do modelo. No entanto, houve unanimidade quanto à utilidade do modelo (muito útil) e os professores que fizeram uso deste recomendam fortemente a sua adoção para outros professores.

Quanto à percepção de aprendizagem dos alunos que adotaram o modelo, o **Gráfico 15** compara os dados coletados com alunos de todo o Brasil no *survey* com os dados obtidos dos grupos do experimento.

Gráfico 15 – Triangulação de Dados entre o *Survey* e o Experimento



Fonte: Elaborado pelo autor.

A partir da análise desse gráfico, verifica-se que a média de percepção de aprendizagem dos alunos que participaram do experimento se aproximou da média dos alunos entrevistados no *survey*, com destaque para o grau "Aprendi muito" do grupo que recebeu o Tratamento 2. No entanto, essa diferença não se mostrou significativa (11% a mais que o controle e 7% a menos que os alunos do *survey*).

Acredita-se que isso se deve à afinidade dos alunos com a área de ES. No questionário aplicado no início do experimento, perguntou-se aos alunos qual era a utilidade da área de ES para a sua formação. Alguns alunos do grupo de Tratamento 1 responderam "Moderadamente útil", enquanto outros do grupo de Tratamento 2 responderam "Completamente inútil". Já todos os alunos do grupo de Controle responderam que a área de ES era "Muito útil" ou "Essencial" para a sua formação. Assim, a variável afinidade pode ter influenciado a motivação dos alunos em se empenhar nas atividades da disciplina e, consequentemente, nos resultados do experimento.

Nesse sentido, destaca-se que larga maioria (89%) dos alunos entrevistados no *survey* considera efetivo para o ensino de ES a realização de Projetos de software, atividades práticas (72%) e discussão de casos práticos (65%). Apenas 43% desses consideram as aulas expositivas (abordagem tradicional) efetivas para a aprendizagem.

5.5 LIÇÕES APRENDIDAS

Segundo a opinião dos professores participantes do experimento, é necessário conhecer as práticas da indústria que são aplicadas, como *coaching* e *mentoring*, para que

se possa fazer uso correto do modelo. Os professores também indicaram que não conseguiram avaliar o grau de restrições para usar o modelo, pois, durante o experimento controlado, o ambiente foi configurado pelos autores deste trabalho. Como lição aprendida, para a próxima avaliação de uso do modelo, os professores usuários deverão instanciar o modelo seguindo apenas as recomendações constantes na sua documentação.

Os autores deste trabalho observaram, durante o experimento, que a variável "motivação" impacta diretamente nos resultados do uso do modelo. Essa mesma observação foi feita por Chen e Chong (2011) em sua experiência de projeto prático de desenvolvimento em ambiente acadêmico. Assim, considera-se que um dos aspectos que o modelo deve instrumentalizar em suas etapas é adoção de técnicas que permitam manter os alunos motivados durante o ciclo de aprendizagem.

Relacionado ao comprometimento, Chen e Chong (2011) também observaram que nem todos os alunos se esforçaram adequadamente, principalmente quando o trabalho e a avaliação são realizados em grupo. Neste sentido, observou-se que os alunos que não possuem afinidade com a área de ES tendem a não realizar as atividades do projeto prático, confiando que os colegas de equipe, mais comprometidos, irão executá-lo. Assim, considera-se que, ao realizar uma avaliação, o professor deva considerar a participação individual de cada membro da equipe do projeto prático.

Outra lição está relacionada com o ensino-aprendizagem de unidades de conhecimento técnicas e não-técnicas. Para cada unidade de conhecimento, observou-se que determinadas etapas do ciclo iterativo do modelo devem ser mais exploradas para desenvolver as competências de forma mais adequada. Por exemplo, para as unidades técnicas de Desenvolvimento de Software, Verificação e Validação e Engenharia de Requisitos, o professor deverá focar nas etapas de Prática e Contextualização. Já para unidades mais gerenciais como Processo de Software, Projeto de Software e Gerenciamento de Projeto, o professor deverá focar nas etapas de Preparação e Discussão.

5.6 CONCLUSÃO

Este capítulo apresentou os resultados da avaliação do modelo de ensino proposto. Inicialmente, avaliou-se a documentação deste modelo através de um painel de especialistas. Os especialistas consideraram que a documentação do modelo estava completa e consistente, mas que possuía baixa usabilidade, falta de clareza e objetividade. A partir deste *feedback*, foram realizadas alterações no modelo a fim de atender as

sugestões de melhoria dos professores especialistas. Dessa forma, o texto do modelo foi revisado e sintetizado para sua versão 2.0. Adicionalmente, desenvolveu-se um sistema *web* a fim de melhorar a sua usabilidade.

Posteriormente, avaliou-se a aplicação do modelo em sala de aula através de um experimento controlado que consistiu na comparação entre duas abordagens de ensino: uma tradicional focada no professor e uma humanista focada no aluno. Assim, definiram-se 3 grupos: controle (abordagem tradicional); tratamento 1 (abordagem humanista – focada no aluno) e tratamento 2 (abordagem humanista – focada no aluno e práticas da indústria). Então, aplicaram-se essas abordagens em duas disciplinas distintas (duas fases) e avaliaram-se a efetividade dessas em relação ao desenvolvimento de competências técnicas nos alunos.

O resultado da Fase 1 do Experimento apresentou diferença significativa entre os 3 grupos, onde tanto o tratamento 1 quanto o tratamento 2 foram mais eficientes que o controle. No entanto, esses resultados não se repetiram na Fase 2 do Experimento, apesar do grupo que seguiu o tratamento 2 apresentar o maior ganho significativo no desenvolvimento de competências técnicas.

A fim de melhor entender esses resultados e reforçar a validade das suas conclusões, realizou-se a triangulação dos dados coletados no experimento controlado com os resultados do painel de especialistas e do *survey*. Assim, no que diz respeito à especificação, o modelo foi considerado adequado e completo para o ensino de ES e a sua documentação correta. No entanto, apesar da melhoria na usabilidade, o modelo continuou não sendo considerado claro e objetivo. A fim de tratar esses pontos fracos, foram realizadas novas alterações no modelo, de acordo com as recomendações dos professores participantes do experimento, atualizando sua versão para a 3.0.

Por fim, no que diz respeito à aplicação do modelo, não se observou diferença significativa na percepção de aprendizagem dos alunos que participaram do experimento em relação aos alunos entrevistados no *survey*. Nesse sentido, acredita-se que a falta de afinidade dos alunos participantes do experimento com a área de ES possa ter influenciado à motivação desses na realização das atividades do experimento. Essa e outras lições aprendidas foram detalhadas nesse capítulo. Já as relações desses resultados com os trabalhos relacionados e suas implicações para a academia e indústria são discutidas no Capítulo 6.

6 DISCUSSÕES

6.1 INTRODUÇÃO

De acordo com a ACM/IEEE (2014), há um consenso de que o ensino de Engenharia de Software no século XXI precisa ir além do formato de leituras para outras variedades de abordagens de ensino-aprendizagem. O ensino com grande concentração de aulas tradicionais pode não ser suficiente para suprir as necessidades de aprendizagem dos alunos (GRILLO, 2003) e, nesse ponto, as diretrizes curriculares da ACM/IEEE (2013) e SBC (2005) indicam a necessidade de aulas que contemplem estratégias de aprendizagem que vão além das aulas expositivas.

Nesse contexto, diversos autores destacam que abordagens práticas parecem ser mais indicadas para o ensino de ES (MALIK e ZAFAR, 2012) (MARQUES, QUISPE e OCHOA, 2014) (SANTOS et al., 2014). Quanto a essas abordagens práticas, observa-se que as mais focadas no aluno e que promovem uma participação mais ativa destes nas aulas têm potencial para aumentar o interesse dos estudantes, motivá-los e melhorar a aprendizagem no nível de aplicação dos conceitos (PRIKLADNICKI et al., 2009).

A partir da análise de mapeamentos sistemáticos (MALIK e ZAFAR, 2012) (MARQUES, QUISPE e OCHOA, 2014) (SANTOS et al., 2014) e de relatos de experiências (OHLSSON e JOHANSSON, 1995) (GNATZ et al., 2003) (PRIKLADNICKI et al., 2009) (ANDRADE et al., 2010) (COUTINHO e BEZERRA, 2010) (MONSALVE, WERNECK e LEITE, 2010) (BESSA, CUNHA e FURTADO, 2012) (GARY et al., 2013), este capítulo apresenta as principais abordagens práticas que vêm sendo adotadas no ensino de Engenharia de Software. Para isso, inicialmente apresenta os trabalhos relacionados a esta pesquisa. Assim, descreve modelos para o ensino com propostas semelhantes à apresentada neste trabalho. Adicionalmente, apresenta enfoques alternativos à adoção desses modelos, como *frameworks* de ensino e técnicas gamificação.

Posteriormente, é realizada uma análise destes trabalhos relacionados, destacando suas vantagens e desvantagens em relação ao propósito desta pesquisa. Por fim, apresentam-se as implicações da adoção do modelo proposto para a academia (ensino e pesquisa) e para a indústria de software (mercado).

6.2 TRABALHOS RELACIONADOS

6.2.1 Modelos para o Ensino de ES

A ES tem sua utilidade reconhecida em grandes projetos, com grandes equipes e orçamentos. Assim, a aprendizagem da área não pode ficar limitada a pequenos projetos irreais, exageradamente simplificados e que ignoram questões de confiabilidade de funcionamento, custos, prazos e outros requisitos importantes (NUNES, REIS e REIS, 2010). No entanto, diversos trabalhos na área de ensino de ES destacam a dificuldade de utilizar por completo, em ambiente acadêmico, vários dos modelos utilizados na indústria, em virtude das particularidades e diferenças entre esses dois tipos de ambientes (RODRIGUES e ESTRELA, 2012) (GIMENES, 2015). Por outro lado, é imprescindível que os alunos façam uso das boas práticas no desenvolvimento de software e se preparem para o mercado, de acordo com as expectativas deste.

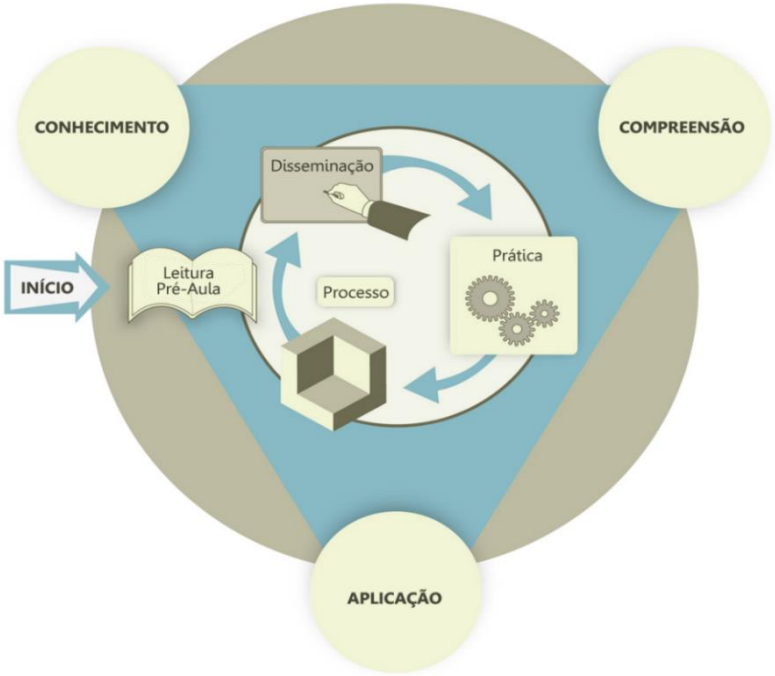
A fim de tratar esta lacuna entre as exigências de formação profissional na área de ES e as limitações do ambiente acadêmico para realização de projetos práticos de desenvolvimento de software, alguns pesquisadores definiram modelos pedagógicos que podem ser adotadas no ensino de ES. A partir desses modelos, os alunos desenvolvem as competências técnicas de maneira mais adequada, auxiliados pelos professores que atuam como tutores neste processo de aprendizado.

Gary (2008) propõe um modelo pedagógico que permite apresentar, praticar e aplicar as melhores práticas da indústria a fim de proporcionar uma aprendizagem mais contextualizada nos alunos. Esse modelo pedagógico possui um currículo flexível e centrado em projeto. O objetivo é integrar experiências contextualizadas de projetos com conceitos fundamentais de ES através de uma abordagem denominada *Software Enterprise*. Essa proposta foi maturada durante 9 anos de aplicação na *Arizona State University Polytechnic*, onde foram desenvolvidos os produtos de trabalho para apoiar a metodologia centrada em projetos (GARY et al., 2013).

A abordagem *Software Enterprise* emprega um modelo de aprendizagem que combina aula tradicional, atividades em grupo, aprendizagem centrada em problema (PBL), prática e reflexão. Inicialmente, os alunos realizam leituras técnicas da disciplina de ES. Em seguida, eles assistem a diversas palestras para criar consciência e compreensão e colocam os conceitos de aula em prática através de sessões de laboratório em grupo a cada semana. Por fim, os alunos integram as habilidades adquiridas durante

o processo para o projeto de software da disciplina de ES, refletem sobre essa rápida experiência de aprendizagem, e se envolvem em grupos de discussão sobre as lições aprendidas. Esse modelo pedagógico é apresentado na **Figura 20**.

Figura 20 – Modelo Pedagógico da Abordagem *Software Enterprise*



Fonte: Adaptado de Gary (2008).

Por exemplo, considerando os tópicos de Gerência de Configuração (GC). Os conceitos de GC como itens de configuração, padrões de *branching* (ramificações) e gestão de mudanças são apresentados em leituras de artigos, feita pelos alunos antes da aula. Durante a mesma semana, a GC é aprofundada via exercícios de laboratório na ferramenta CVS. Na semana seguinte, a GC é incorporada ao projeto prático da disciplina. Ao final de uma iteração de 3 semanas, as equipes devem escrever em uma *wiki* instruções para aplicação de técnicas no projeto e uma avaliação do quão bem a abordagem funcionou.

Esse modelo busca facilitar a compreensão através da aplicação de conhecimento e, consequentemente, preparar melhor os graduandos para o mercado de trabalho (GARY et al., 2013). Além disto, esse modelo baseia-se no ciclo de aprendizagem de Kolb (1984) apresentado na Subseção 2.4.3 - A.

Gary (2008) destaca que o acoplamento entre o conhecimento adquirido e as habilidades práticas aplicadas nas tarefas do processo leva à compreensão mais rápida e

a uma aprendizagem mais profunda do que o modelo tradicional. O autor classifica esse modelo pedagógico como um “modelo iterativo centrado no aluno e facilitado por um instrutor”. No entanto, reforça que utilizar uma metodologia de ensino/aprendizagem altamente iterativa coloca uma grande responsabilidade sobre os instrutores/facilitadores. Exercícios práticos para aprendizagem centrada no problema devem ser construídos. Os facilitadores devem determinar a quantidade correta de orientação e apoio que devem fornecer às equipes dos projetos que viabilize a aprendizagem sem gerar forte influência nas atividades, desviando os alunos de encontrar por si mesmos o “caminho certo”. Por fim, Gary (2008) enfatiza que os instrutores devem repensar o modo como avaliam o aprendizado e como avaliam o relativo sucesso do projeto prático.

Quanto à avaliação, Gary et al. (2012) constataram em suas pesquisas que o mais importante é criar uma base conceitual duradoura pela qual novas experiências são assimiladas para aumentar a base de conhecimento pessoal do aluno de uma forma ordenada. Dessa forma, adotaram em sua abordagem de *Software Enterprise* mapas conceituais como uma técnica para avaliar a evolução da compreensão do aluno em relação ao conhecimento conceitual em Engenharia de Software. O protocolo de avaliação de mapas conceituais segue as seguintes etapas:

1. No início do semestre, os estudantes criam um mapa conceitual, referente a uma unidade de conhecimento, a partir de um *template* com os conceitos dessa unidade;
2. Os estudantes são instruídos a organizar os conceitos nos mapas da maneira que melhor reflita o seu entendimento sobre a unidade de conhecimento;
3. Ao final do semestre, os estudantes criam outro mapa conceitual, a partir do mesmo *template* e das mesmas instruções. Eles não podem consultar o mapa criado no início do semestre.

Então, avaliadores especialistas criam mapas conceituais para cada unidade de conhecimento como gabarito. Em seguida, esses avaliam e ranqueiam os pré e pós-mapas dos alunos de acordo com esse gabarito. Assim, pode-se avaliar o quanto o mapa de um aluno se aproximou do mapa do especialista e a trajetória de aprendizado do aluno durante o semestre.

O modelo pedagógico de Gary et al. (2013) serviu como base para instrumentalização da abordagem iterativa do modelo proposto nesta tese. O referido trabalho auxiliou na definição das etapas, bem como na ordem de realização destas. Além

disso, adotou-se na avaliação de aprendizagem dos alunos, durante a realização do experimento controlado (ver Seção 5.3), a abordagem de mapas conceituais a fim de analisar a ordenação de conceitos dos alunos e a sua trajetória de aprendizado.

Apesar da contribuição, esse trabalho possui algumas limitações. Os autores não fizeram uma análise da correlação do conteúdo abordado pelo modelo com os currículos de referência da área. Isso faz com que não seja possível afirmar que se de fato o modelo aborda os tópicos da área de ES. Adicionalmente, o modelo é aplicado em uma disciplina específica, denominada *Software Enterprise*, e com o conteúdo focado em Linguagens de Programação, Algoritmos e Estrutura de Dados, Programação *Web* e *Mobile*.

O diferencial do modelo proposto nessa pesquisa consiste na adoção de tópicos dos currículos de referência considerados relevantes pelos professores entrevistados no *survey* (ver Seção 3.5). Assim, na instanciação do modelo, o professor poderá definir qual unidade de conhecimento de ES seguirá a abordagem iterativa (ver Subseção 4.4.1). Essa abordagem permite que o modelo possa ser aplicado em qualquer disciplina da área de ES.

Por fim, destaca-se que o modelo de Gary et al. (2013) não foi avaliado através de um método empírico de pesquisa. Sendo assim, ainda se realizou uma análise da eficiência da sua aplicação. O modelo apresentado nesta tese foi aplicado em 2 turmas de graduação, através de um experimento controlado, sendo os dados analisados de forma qualitativa e quantitativa.

6.2.2 *Frameworks* de Ensino

De acordo com Caspersen e Christensen (2008), há um conceito geral para o termo "*framework*" que define uma estrutura conceitual básica ou um quadro estrutural. Nesse sentido, os autores destacam que um *framework* especifica: um comportamento em alto nível de abstração; um domínio bem definido de atuação; padrões de interação; uma aplicação flexível, de modo que você pode adaptá-lo a um contexto concreto (contanto que esse contexto se encontre dentro do domínio do *framework*). Esse conceito foi aplicado na área do ensino de Engenharia de Software por Hazzan e Dubinsky (2006) e por Sowe, Stamelos e Deligiannis (2006).

Hazzan e Dubinsky (2006) definiram um *framework* para o ensino de Métodos de Desenvolvimento de Software (MDS) baseado em aprendizagem ativa. Esse *framework*

é baseado em 10 (dez) princípios conceituais e especifica 5 (cinco) práticas referentes às atividades de ensino que orientam os professores na avaliação dos estudantes bem como no processo de desenvolvimento da disciplina de ES. A **Tabela 15** apresenta esses princípios seguidos de uma breve descrição.

Tabela 15 – 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 e Dubinsky (2006).

O Princípio I aborda a estrutura do curso, enquanto os Princípios de II a V tratam do ensino real dos MDSs. Já os Princípios de VI a IX tratam das pessoas envolvidas no ambiente educacional – os estudantes e a equipe de professores. Por fim, o Princípio X é um meta-princípio que sugere a ação ao invés da imposição. Enquanto os dez princípios são ideias conceituais em que o *framework* se fundamenta, as cinco práticas abordam a implementação concreta do *framework*.

Quanto a essas práticas, a primeira é a Prática de Curso, que implementa a estrutura do curso considerando uma perspectiva cognitiva. As ferramentas usadas na construção dessa prática foram vídeos de reuniões de projetos, reflexões de alunos, mensagens em fóruns, questionários e entrevistas com estudantes das equipes do projeto. Já a segunda, Prática de Papéis, considera todos os estudantes como desenvolvedores e, adicionalmente, cada membro da equipe possui um papel especial de acordo com um aspecto do gerenciamento do projeto de software. Esses papéis foram definidos de acordo com uma adaptação dos papéis do *eXtreme Programing* (XP) para o contexto acadêmico.

A terceira é a Prática de Medição, que especifica como dados quantitativos são usados para derivar medidas. Foram utilizadas 3 (três) medidas: a) Medida de Tempo no Papel, que mede a relação entre o tempo investido em tarefas de desenvolvimento e o tempo investido em atividades de seu papel específico; b) Medida de Comunicação, que mede o nível de comunicação da equipe em cada fase do desenvolvimento; c) Medida de Gerenciamento, que mede o nível de gerenciamento do projeto através da performance dos papéis.

A quarta é a Prática de *Coaching*, que consiste em analisar os resultados de treinamentos realizados por professores. A partir de questionários, os instrutores apresentam suas perspectivas e concepções sobre o tema do treinamento, como, por exemplo, sobre XP. Por fim, a quinta é a Prática de Avaliação, que avalia o trabalho desenvolvido pelos alunos. Quanto à avaliação, 65% é baseada na reunião de apresentação das estórias para os usuários bem como nas estimativas dos estudantes para cada uma das 3 iterações. Os demais 35% dizem respeito à avaliação individual do estudante em relação às tarefas desenvolvidas e ao seu papel no projeto.

De acordo com Hazzan e Dubinsky (2006), existem relações mútuas entre essas práticas. Mais especificamente, cada prática contribui para o desenvolvimento e formulação de outras práticas. Por exemplo, a Prática de Avaliação influencia o cronograma e as medidas de desenvolvimento que fazem parte do curso. Já a Prática de Medição relaciona-se com o papel do estudante, que faz parte da Prática de Papéis, e ao *feedback* dos *coache*s acadêmicos que fazem parte da Prática de *Coaching*.

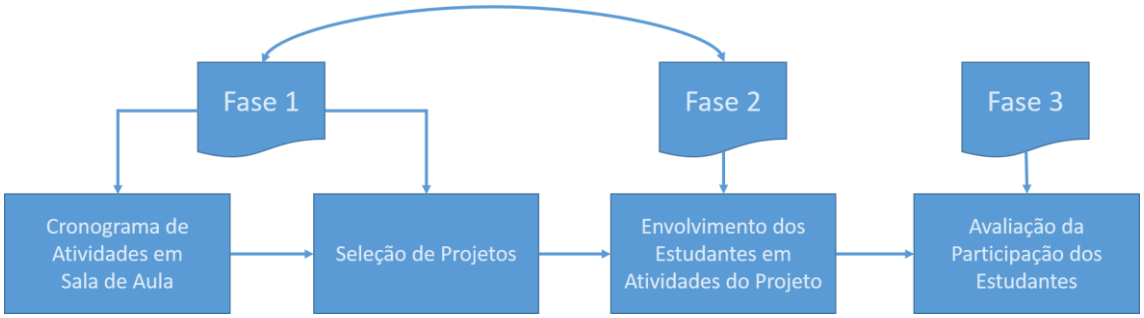
O *framework* proposto por Hazzan e Dubinsky (2006) é bem compreensível e detalha como implementá-lo em cursos de graduação. O modelo proposto nesta pesquisa se baseará na sua estrutura conceitual, definindo princípios e práticas. No entanto seu foco é no ensino de MDS. Como diferencial, o modelo proposto pode ser implementado no ensino de qualquer tópico das unidades de conhecimento da ES.

Adicionalmente, esse modelo fornece um sistema web que facilita a instanciação das etapas do modelo e disponibiliza uma base de material de apoio, o que aumenta a sua usabilidade em relação ao *framework* de Hazzan e Dubinsky (2006). Assim, o professor pode optar por adotar integralmente a sintaxe do modelo ou optar por adotar somente etapas específicas, tais como: PBL, jogos, dinâmicas artigos ou videoaulas.

Outro *framework* foi proposto na área por Sowe, Stamelos e Deligiannis (2006), que usa a metodologia de desenvolvimento de software *open source* e foi implementado na disciplina de "Introdução à Engenharia de Software" no departamento de Informática da *Aristotle University* na Grécia. Em projetos de software livre e *open source*, os desenvolvedores compartilham o código-fonte com uma grande quantidade de testadores e garantem uma rápida evolução neste, pois *bugs* e defeitos são identificados e resolvidos mais rapidamente (SOWE, STAMELOS e DELIGIANNIS, 2006).

Dessa forma, o *framework* proposto por Sowe, Stamelos e Deligiannis (2006) possui 3 (três) fases, onde cada uma descreve um contexto de ensino e aprendizagem em que os estudantes são envolvidos em atividades de um projeto de software real. A **Figura 21** representa essas fases do *framework* e o fluxo entre elas.

Figura 21 – *Framework* para Desenvolvimento de Software *Open Source*



Fonte: Adaptado de Sowe, Stamelos e Deligiannis (2006).

Na Fase 1, os estudantes realizaram 8 horas de leitura, no início do semestre, sobre os tópicos:

- Atividades e testes em projetos de software livre e *open source*;
- Formação e papéis em comunidades de software livre e *open source*;
- Comunicação através de fóruns e listas de e-mails;
- Plataformas colaborativas como CVS, Bugzilla, dentre outros.

Após a fase de leitura, os estudantes são guiados para projetos hospedados no repositório do *sourceforge.net*. Nesse repositório, os estudantes escolhem um projeto e fazem uma apresentação, detalhando o histórico desse projeto, o procedimento para relatar *bugs* e as ferramentas de testes utilizadas.

Na Fase 2, os estudantes aprendem a registrar os seus projetos em ferramentas de rastreamento de *bugs*. Inicialmente, eles escrevem *bugs* fictícios para serem analisados e criticados pelos seus colegas de turma. Então, após essa etapa inicial, os estudantes

iniciam os testes nos softwares livres e *open source* dos repositórios do *sourceforge.net* seguindo os seguintes passos: (1) *Download* e instalação do software; (2) Aplicação de várias técnicas de teste; (3) Descoberta de *bugs* que são registrados na base de dados do projeto seguindo o procedimento de reporte; (4) Se o estudante não encontra *bug*, ele pode executar mais testes ou testar outro projeto; (5) Toda vez que um estudante submete um *bug*, ele notifica um observador na turma; (6) Os estudantes são solicitados a fazerem uma apresentação na turma para discutir suas experiências.

Na Fase 3, baseado na apresentação dos alunos e nas atividades de teste, os estudantes são avaliados: (10%) apresentação em turma; (12%) participação no projeto; (13%) relatório de *bugs*; (15%) atividades de teste. Os demais 50% da avaliação são destinados ao restante dos tópicos da disciplina.

Uma abordagem semelhante foi adotada por Fox (2015) em uma disciplina de ES, onde os alunos escolhem um "*software as a service*" legado na *Cloud Computing* para aprimorar. Ao desenvolver um novo aplicativo que corresponda aos requisitos de clientes não-técnicos, os alunos experimentam todo o ciclo de vida do software durante a disciplina e usam habilidades que a indústria demanda.

A principal contribuição do *framework* de Sowe, Stamelos e Deligiannis (2006) para o modelo proposto é a exposição dos estudantes a projetos reais de ES e a divisão do modelo em fases bem distintas. Entretanto, esse *framework* aborda apenas a área de testes. Destaca-se que, a partir da realização do experimento controlado, foi possível observar que o modelo proposto pode ser implementado no ensino das diversas unidades de conhecimento da ES. Sendo assim, não se limita ao ensino de testes de software. No experimento controlado, por exemplo, o modelo foi usado no ensino de Engenharia de Requisitos e Gerenciamento de Projetos.

A interação dos alunos com projetos reais, na abordagem de Sowe, Stamelos e Deligiannis (2006), é realizada somente através da participação em fóruns e do desenvolvimento colaborativo de softwares livres. O modelo proposto propõe uma abordagem mais próxima, através do convite de profissionais para palestrar e discutir sobre sua área de atuação (ver Subseção 4.4.3), de acordo com a unidade de conhecimento ministrada na disciplina. Adicionalmente, recomenda-se o uso de problemas e clientes reais na etapa de Contextualização (ver Subseção 4.4.5).

6.2.3 Gamificação

Tecnologias recentes têm proporcionado novas oportunidades para o uso de jogos, bem como para a aplicação dos seus elementos para melhorar a aprendizagem e o envolvimento dos alunos. Destaca-se nesse cenário o uso recente na ES do método *Gamification* (Gamificação) que, de acordo com Deterding et al. (2011), consiste na aplicação de elementos e estratégias de jogos virtuais no ensino. O objetivo desse método é engajar e motivar os alunos para que eles assimilem o conhecimento, não somente em sala de aula, mas em qualquer lugar.

O que a gamificação propõe, como estratégia aplicável aos processos de ensino e aprendizagem nas escolas ou em qualquer outro ambiente de aprendizagem, é utilizar um conjunto de elementos comumente encontrados na maioria dos *games* e aplicá-los nesses processos, com o intuito de gerar níveis semelhantes de envolvimento e dedicação daqueles que os *games* comumente conseguem gerar (FARDO, 2013).

Nesse sentido, de acordo com Zichermann e Cunningham (2011), pode-se utilizar da mecânica de jogos, dinâmicas e estéticas em um ambiente que não seja o próprio jogo. A Dinâmica é a interação do usuário com o ambiente por meio das Mecânicas. Uma Dinâmica envolvente é capaz de impulsionar um sistema gamificado, e, ao contrário das Mecânicas, que são definidas pelos *designers*, é fruto da ação dos jogadores sobre as regras. Já a Estética é a emoção que os jogadores estão sentindo durante a interação com o ambiente e, necessita ser positiva.

De acordo com Fardo (2013), a gamificação também se dispõe a transpor os métodos de ensino e aprendizagem presentes nos *games* para a educação formal. Em outras palavras, aplicar a gamificação em um determinado contexto significa observá-lo e propor soluções sob a perspectiva de um *designer* de games, especificamente naquilo que ele faria com o mesmo problema e quais as estratégias que utilizaria caso esse problema fosse o tema de um *game*, em um mundo virtual (com todas as restrições e cuidados éticos e metodológicos que o mundo real implica).

Assim, Zichermann e Cunningham (2011) destacam as seguintes estratégias que podem ser adotadas no processo de ensino-aprendizagem:

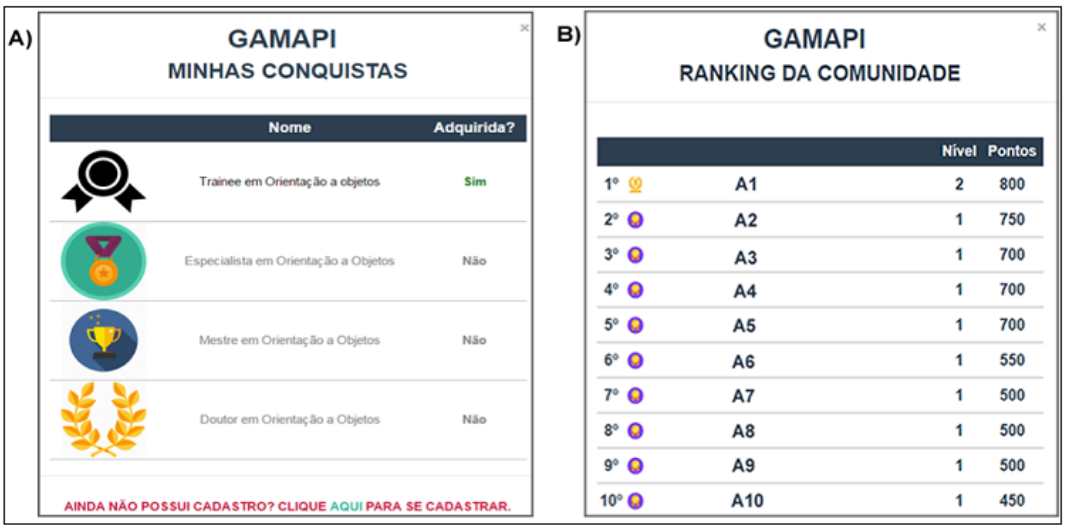
- **Pontos:** permitem quantificar o desempenho do aluno, auxiliando-o a estabelecer metas e objetivos;

- **Níveis:** transmitem o progresso do jogador dentro do sistema, determinando o grau de competência que um aluno possui dentro do ambiente;
- **Conquistas:** são recompensas dadas quando um jogador alcança determinada pontuação e podem estar associadas a um Emblema;
- **Emblemas:** são formas de transmitir *status*, pois são um meio visual de representar as vitórias alcançadas. Podem ser usados como desafios extras, a fim de direcionar o aluno a um comportamento desejado;
- ***Rankings*:** são usados para expor os ganhos do usuário perante a comunidade a qual ele pertence, concebendo um meio de transmitir um incentivo social. É uma maneira de influenciar outros indivíduos, que não estão no topo do *ranking*, a almejar essa posição despertando o espírito competitivo;
- ***Feedback*:** tem uma função fundamental nos ambientes gamificados, uma vez que auxilia a manter o usuário devidamente informado sobre o que está acontecendo no sistema.

Boas et al. (2017) definiram uma *Application Programming Interface* (API) para Gamificação, denominada GamAPI, que objetiva proporcionar uma opção para implementar os conceitos e os mecanismos de gerenciamento de gamificação sem a necessidade de se ter uma estrutura própria e que conduza o aluno a uma experiência prazerosa de ensino.

A **Figura 22** demonstra a interatividade do GamAPI em um ambiente de ensino por meio das conquistas do usuário (item A) e do *ranking* de usuários (item B).

Figura 22 – Exemplos de Recursos de *Feedback* do GamAPI



Fonte: Boas et al. (2017).

No item A é possível visualizar as conquistas do usuário, bem como aquelas que ainda não alcançou. Essa progressão busca ajudar a manter o usuário no sistema, pois fornece meios para que ele possa estabelecer seus objetivos e, desse modo, tende a mantê-lo motivado. Já o item B exibe o *ranking* dos usuários por pontuação. Também são apresentados seus níveis e os emblemas correspondentes a esses. Esse tipo de mecânica, quando inserida dentro de uma rede social, busca transmitir *status*, despertar o espírito competitivo e permitir aos usuários analisar suas próprias performances.

Além desses recursos do GamAPI, os autores destacam: o gráfico comparativo de progressão entre os indivíduos, que dispõe informações relacionadas às conquistas cadastradas no sistema, e o que cada um conseguiu desbloquear; um resumo que exibe o que os indivíduos conquistaram até o momento, como os níveis alcançados, a quantidade de pontos adquiridos e as conquistas desbloqueadas. Esse *feedback* imediato e em tempo real busca propiciar a percepção de imersão no sistema da API.

A fim de validar o GamAPI, foi realizado um quasi-experimento com 10 alunos do curso de Graduação em Engenharia da Computação da Universidade Tecnológica Federal do Paraná. Sendo assim, o professor criou uma comunidade no GamAPI, com o tema Tópicos em Engenharia de Software, na qual foram cadastradas, inicialmente, 4 conquistas e 2 níveis. As conquistas cadastradas pelo professor foram: Trainee em Orientação a Objetos, com 100 pontos necessários para o seu desbloqueio; Especialista em Orientação a Objetos, com 250 pontos; Mestre em Orientação a Objetos, com 500 pontos e, Doutor em Orientação a Objetivos, com 800 pontos.

Os dados adquiridos a partir da análise dos resultados do quasi-experimento demonstraram a efetividade do uso da GamAPI, pois os alunos interagiram positivamente com as funcionalidades da API (BOAS et al., 2017). Os autores destacam que houve, durante a aplicação do teste, uma demonstração do anseio dos alunos pelas posições superiores do *ranking*. Afirmam ainda que foi notada certa euforia por parte desses alunos ao conseguirem desbloquear as conquistas ou evoluir de nível. Nesse caso, a API funcionou corretamente ao promover a interação com o usuário em tempo real, e, em conjunto com as técnicas de gamificação, influenciou positivamente os alunos na resolução de suas atividades.

Apesar de apresentar resultados expressivos, ainda existem poucos relatos da aplicação de gamificação no ensino de ES na literatura. Por consequência, há poucos materiais de referência para criação de mecânicas de jogos para o ensino de unidades

específicas da ES. Adicionalmente, destaca-se a complexidade em se criar essas mecânicas, pois é necessário que o professor tenha habilidades de *game designer*.

Essa abordagem possui uma grande complexidade associada à criação de mecânicas específicas para o ensino de cada unidade de conhecimento da ES. Diferentemente dessa abordagem, o modelo aqui proposto permite ao professor lecionar diversas unidade da ES de forma integrada por meio um ciclo iterativo, composto por 6 (seis) etapas.

6.3 RELAÇÃO COM TRABALHOS RELACIONADOS

6.3.1 Comparação com Modelos Adotados no Ensino de ES

Gary et al. (2013) apresentaram um modelo iterativo centrado no aluno e facilitado por um instrutor. Esse modelo, baseado no ciclo de aprendizagem de Kolb (1984), visa permitir que os alunos apliquem o conhecimento e, consequentemente, preparem-se melhor para o mercado de trabalho. Permite apresentar, praticar e aplicar as práticas da indústria e proporcionar uma aprendizagem dos conceitos de ES de forma mais contextualizada.

No entanto, os autores destacam a dificuldade de preparar o material de apoio para o uso em um ambiente de aprendizado colaborativo orientado à prática, bem como preparar as atividades para aprendizagem centrada no problema. Adicionalmente, citam que os professores devem ter cuidado com a quantidade de apoio que fornecem aos alunos para que não prejudiquem a sua aprendizagem, pois é preferível que esses encontrem as soluções e respostas por si mesmos.

Nesse sentido, a contribuição da presente pesquisa se dá através da disponibilização de uma base de materiais de ensino, disponível no sistema *web* do modelo, onde professores podem baixar e submeter suas aulas, compartilhar artigos, vídeos, jogos e dinâmicas, dentre outros. Além disso, o modelo disponibiliza uma documentação que especifica o passo-a-passo e exemplos de intervenções que o professor pode fazer nas atividades de ensino-aprendizagem.

Por fim, destaca que os resultados obtidos no experimento controlado permitiram reforçar as conclusões da pesquisa de Prikladnicki et al. (2009) no que diz respeito à efetividade do ensino centrado nos alunos. Os professores participantes destacaram que os alunos que participaram do grupo que seguiu abordagens práticas foram bem mais

participativos e proativos, sendo as aulas guiadas a partir das suas iniciativas ao invés do tradicional seguimento da ementa da disciplina.

6.3.2 Comparação com *Frameworks* para o Ensino de ES

O *framework* definido por Hazzan e Dubinsky (2006), baseado em princípios conceituais de aprendizagem ativa, apoia o ensino de Métodos de Desenvolvimento de Software (MDS). Já o *framework* proposto por Sowe, Stamelos e Deligiannis (2006), baseado na metodologia de desenvolvimento de software *open source*, apoia o ensino de testes de software.

O modelo proposto incorporou o conceito de flexibilidade destes *frameworks*, permitindo que seus usuários possam escolher quais etapas irão adotar no ensino de ES. No entanto, como diferencial, esse modelo fornece um sistema *web* que facilita essa instanciação das etapas do modelo e disponibiliza uma base de material de apoio, o que aumenta a sua usabilidade em relação aos *frameworks* identificados na literatura. Assim, o professor pode optar por adotar integralmente a sintaxe do modelo ou optar por adotar somente etapas específicas, tais como: PBL, jogos, dinâmicas artigos ou videoaulas.

Assim como o *framework* de Hazzan e Dubinsky (2006), o modelo proposto definiu princípios e práticas e, como Sowe, Stamelos e Deligiannis (2006), colocou os estudantes em contato com profissionais da indústria a fim de que esses pudessem ouvir relatos de experiências e tirar dúvidas sobre questões técnicas. Como diferencial, destaca-se que, a partir da realização do experimento controlado, foi possível observar que o modelo proposto pode ser implementado no ensino das diversas unidades de conhecimento da ES. Sendo assim, não se limita ao ensino de MDS ou testes de software. No experimento controlado, o modelo foi usado no ensino de Engenharia de Requisitos e Gerenciamento de Projetos.

Assim, pode-se concluir que o modelo iterativo apresentado nesta tese se mostra mais adequado para o ensino de uma disciplina da área de ES, já que não se limita a uma unidade ou tópico de conhecimento como os *frameworks* apresentados na Subseção 6.2.2. Através do seu ciclo iterativo, é possível que o professor possa dar continuidade na realização de um projeto prático após a finalização do ciclo de uma unidade. Por exemplo, o professor pode usar o modelo para lecionar Engenharia de Requisitos e em seguida realizar o Gerenciamento do Projeto.

6.3.3 Comparação com Estratégias de Gamificação

A Gamificação busca engajar e motivar os alunos a fim de que esses assimilem o conhecimento através de estratégias comuns de *games.* Nesse contexto, observa-se o uso recente de mecânica de jogos, dinâmicas e estéticas no ensino de ES. Boas et al. (2017) aplicaram técnicas de gamificação em uma disciplina de ES, destacando como resultados a competição entre os alunos pelas posições superiores no *ranking* e o aumento de grau de competência ao conseguirem desbloquear as conquistas ou evoluir de nível.

Apesar dos diversos benefícios relatados em relação à adoção de gamificação, destaca-se a complexidade em se criar mecânicas específicas para o ensino de cada unidade de conhecimento da ES. Ainda existem poucos relatos disponíveis na literatura sobre o uso de gamificação no ensino destas unidades da ES. Diferentemente dessa abordagem, o modelo aqui proposto permite ao professor lecionar diversas unidade da ES de forma integrada por meio um ciclo iterativo, composto por 6 (seis) etapas.

No entanto, visando obter os benefícios inerentes à gamificação, recomenda-se que o professor que adotar o modelo proposto nessa tese possa inserir práticas como pontos, *ranking*, conquistas e *feedback*. Os pontos e *ranking* podem ser adotados para motivar e engajar os alunos nas fases de Prática (durante a realização de dinâmicas ou uso de jogos educativos) e Contextualização (na avaliação dos produtos de trabalho produzidos). Para o aluno que obtiver a melhor classificação ou para os membros da equipe que obtiverem a melhor média, o professor pode recompensá-lo(s) através de conquistas, como um *template* de documentação preenchido para que o aluno possa se basear no projeto prático. Por fim, quanto ao *feedback*, os alunos devem fornecer aos professores durante as atividades de *coaching* e o profissional deve fornecer aos alunos durante as atividades de *mentoring*.

De acordo com Chen e Chong (2011), motivar os alunos é um dos principais desafios para os professores da área de ES. A fim de tratar a variável “motivação dos alunos”, pretende-se, na próxima versão, adotar estratégias de gamificação em todas as etapas do modelo. Na etapa de Iniciação, pode-se definir o ambiente e contexto do jogo. Na Preparação, pode-se recompensar os alunos pela leitura dos artigos. Na etapa de Discussão, o profissional pode fornecer *feedback* sobre as decisões técnicas dos alunos. Posteriormente, na etapa de Prática, pode-se utilizar as estratégias de níveis e pontos nos jogos educativos e dinâmicas. Já na Contextualização, pode-se gerar um *ranking* a partir

da avaliação dos produtos de trabalho entregues. Por fim, na Reflexão, o professor pode fornecer um *feedback* elaborado sobre o ciclo de ensino-aprendizagem e premiar as equipes com emblemas, por exemplo.

6.4 IMPLICAÇÕES PARA A ACADEMIA

As principais implicações dos resultados desta pesquisa se dão no contexto acadêmico, tendo em vista que o público-alvo do modelo são professores de disciplinas da área de ES. Nesse sentido, espera-se que os professores sejam orientados na seleção de técnicas, estratégias e métodos de ensino adequados para motivar e engajar os alunos. Consequentemente, o modelo beneficia os alunos das turmas que o seguirem, uma vez que este busca desenvolver competências técnicas para aqueles que pretendem atuar em cargos relacionados à ES. De acordo com Lethbridge et al. (2007), alinhar o ensino de ES com as necessidades da indústria ajuda a diminuir uma das lacunas mais críticas da área.

Tradicionalmente, a maioria dos currículos de graduação foca no conteúdo. O modelo proposto foca no ensino desse conteúdo baseado em problemas e projetos que os alunos enfrentarão na prática. Dessa forma, espera-se que os professores que adotarem o modelo possam ajudar a alterar a atual dinâmica de ensino de ES, tornando-a mais atrativa aos alunos. De modo indireto, espera-se desenvolver durante a formação profissional as competências técnicas requeridas pela indústria através do seguimento de abordagens focadas nos alunos e práticas de capacitação da indústria adaptadas para uso no contexto acadêmico.

Adicionalmente, destaca-se como resultado acadêmico a criação de um grupo de pesquisa denominado Laboratório de Abordagens de ensino Focadas no Aluno (LA FocA) sob portaria nº 007/2017 na Universidade Federal do Pará (UFPA), Campus Universitário do Tocantins/Cametá. Esse grupo possui 16 membros, sendo 2 professores coordenadores, 4 professores colaboradores, 6 alunos da pós-graduação (especialização) e 6 alunos da graduação.

O LA FocA está registrado no Diretório dos Grupos de Pesquisa no Brasil (GDP) do CNPq, acessível pelo endereço *dgp.cnpq.br/dgp/espelhogrupo/3604162452120260*. Destaca-se como resultado desse grupo a aprovação do Projeto de Extensão denominado “GamEFun – Gamificação no Ensino Fundamental”, sendo contemplado com 2 (duas) bolsas de monitoria e 1 (uma) de extensão.

6.5 IMPLICAÇÕES PARA A INDÚSTRIA

Os resultados deste trabalho possuem implicações para a indústria de software, tendo em vista que o modelo proposto pretende desenvolver competências técnicas em nível de aplicação, impactando diretamente a qualidade dos profissionais. Ao adotar práticas de capacitação consideradas eficientes no desenvolvimento de competências técnicas, os resultados dessa pesquisa podem, em longo prazo, contribuir para o desenvolvimento de produtos de software com a qualidade esperada pelos usuários.

Destaca-se que essas práticas são derivadas de estratégias de consultores que implementam modelos de qualidade, como o MR-MPS-SW e CMMI-DEV. Assim, as empresas que contratarem profissionais com conhecimento prévio dessas práticas, mesmo que apenas em sua formação acadêmica, tendem a se tornar mais competitivas no mercado nacional e internacional.

Outra implicação esperada dos resultados desta pesquisa diz respeito ao aumento da colaboração academia-indústria a partir do convite/participação de profissionais em disciplinas da área de ES. De acordo com Mead (2009), esse tipo de colaboração permite a realização de pesquisas relevantes e o aprofundamento da experiência prática dos educadores. Espera-se que esses profissionais possam compartilhar suas experiências de mercado e, eventualmente, motivar e serem motivados pelos alunos, contribuindo assim para a formação desses.

Por fim, destaca-se que o modelo está disponível para instanciação, podendo ser usado e customizado para o treinamento de equipes de profissionais. Treinadores que seguirem as etapas do modelo poderão atualizar suas estratégias de capacitação, através do uso de jogos e gamificação por exemplo, e/ou contribuir com o fornecimento de materiais de apoio ao ensino de unidades de ES.

6.6 CONCLUSÃO

Este capítulo apresentou diversas discussões sobre os resultados obtidos e as implicações destes. Inicialmente, para situar essa pesquisa na área de ensino de ES, apresentaram-se trabalhos relacionados que contribuíram diretamente tanto na fundamentação da tese quanto na definição da proposta de modelo. Dentre esses, destaca-se os trabalhos de Gary (2008) e Gary et al. (2012) (2013) que definiram um modelo pedagógico para o ensino de ES. Adicionalmente, apresentaram-se trabalhos que possuem

objetivos semelhantes aos apresentados nesta pesquisa e que podem ser adotados para o ensino prático de ES. Nesse contexto, destacaram-se os *frameworks* que definem uma estrutura conceitual básica com fases bem definidas e aplicação flexível para o ensino de ES, e a gamificação, que permite engajar e motivar os alunos a partir de práticas de jogos adaptadas para o contexto de ensino.

Diferente dos trabalhos relacionados apresentados, o modelo iterativo para o ensino proposto nesse trabalho pode ser aplicado a qualquer unidade de conhecimento da ES e abrange os diferentes perfis de aprendizagem dos alunos. Além disso, incorpora a fundamentação de 7 (sete) princípios para o ensino prático de ES e alguns dos benefícios dos trabalhos relacionados apresentados nesse capítulo.

Por fim, apresentaram-se as principais implicações do modelo proposto para a academia e para a indústria. No contexto da academia, espera-se que os professores que adotarem o modelo possam selecionar técnicas, estratégias e métodos de ensino adequados para motivar e engajar os alunos. No contexto da indústria, espera-se que o modelo proposto impacte diretamente a qualidade dos profissionais através do desenvolvimento de competências técnicas em nível de aplicação.

7 CONSIDERAÇÕES FINAIS

7.1 SUMÁRIO DO TRABALHO

De maneira geral, a indústria de software apresenta insatisfação quanto ao nível de preparação dos profissionais recém-formados (WANGENHEIM e SILVA, 2009). Especificamente, as empresas se queixam de que os cursos de graduação não ensinam aos estudantes as competências e as habilidades profissionais necessárias para que esses possam começar a executar suas atividades com eficiência (BESSA, CUNHA e FURTADO, 2012). Dessa forma, as empresas de software investem em treinamentos e certificações desses profissionais recém-formados para repasse técnico de práticas específicas da área de Engenharia de Software (MEIRA, 2015).

Essa carência na formação de profissionais graduados na área de ES pode ser resultado de uma educação inadequada (LETHBRIDGE et al., 2007). Particularmente nos cursos de graduação, os tópicos de ES são normalmente ensinados de forma bastante superficial, pois há muito conteúdo a ser ministrado em pouco tempo (WANGENHEIM e SILVA, 2009) e professores que não possuem experiência em determinadas unidades de conhecimento. Além disso, muitos professores priorizam aulas didáticas/teóricas que acabam por desmotivar os estudantes no estudo dos conceitos da ES (PRIKLADNICKI et al., 2009), consequentemente, diminuindo o comprometimento com o aprendizado. Somam-se a esses fatores as dificuldades em preparar em ambientes acadêmicos os estudantes para a prática profissional fora deles (RODRIGUES e ESTRELA, 2012).

Diante desse contexto, acredita-se que, se as disciplinas da área adotassem métodos de ensino focados no aluno e práticas de capacitação da indústria, se poderia desenvolver as competências e habilidades profissionais de maneira mais adequada. Por consequência, isso poderia reduzir o déficit de profissionais qualificados e a quantidade de tempo e investimentos gastos em capacitação de profissionais recém-formados. Sendo assim, este trabalho propôs um modelo de ensino que incorpore essas características.

Antes de se identificar esse modelo como uma possível solução para as problemáticas expostas, várias etapas desta pesquisa foram conduzidas. Inicialmente definiram-se os objetivos e questões de pesquisa (Seção 1.3) e realizou-se uma revisão da literatura. Essa revisão permitiu construir a fundamentação teórica desse trabalho e possibilitou a identificação dos trabalhos relacionados a esse (Seção 6.2).

Em seguida, um *survey* (Seção 3.5) foi conduzido com professores e estudantes concluintes das disciplinas da área de ES. Esse *survey* permitiu identificar os tópicos de ES mais adotados pelos professores e as abordagens de ensino consideradas mais efetivas pelos estudantes. Além disso, permitiu reforçar as problemáticas identificadas na literatura especializada.

A fim de relacionar os tópicos mais adotados pelos professores com as melhores práticas da indústria de software realizou-se um mapeamento (Seção 3.6) entre os tópicos de ES e as recomendações dos modelos de qualidade CMMI-DEV (SEI, 2010) e MR-MPS-SW (SOFTEX, 2012). A partir desse mapeamento, foi conduzido um levantamento com consultores em MPS das estratégias de capacitação adotadas por esses na indústria para desenvolver competências profissionais.

Finalmente, optou-se por integrar as práticas de capacitação adotadas pela indústria e as abordagens práticas para o ensino em um modelo iterativo (Capítulo 4) que potencializa os benefícios da aplicação conjunta das mesmas. Esse modelo é composto por 6 (seis) etapas que definem a sua sintaxe. Adicionalmente, define um sistema social que descreve seus componentes e as interações entre eles. Por fim, especifica os requisitos e materiais de apoio necessários para sua aplicação em sala de aula.

A documentação do modelo foi avaliada através de um painel de especialistas (Seção 5.2). Esses especialistas consideraram o modelo adequado e completo para o ensino de ES. Para a avaliação da efetividade do modelo, realizou-se um experimento controlado (Seção 5.3) em duas turmas de graduação a fim de comparar a sua aplicação em sala de aula com uma abordagem tradicional de ensino. Os resultados obtidos para a unidade de Gerenciamento de Projetos demonstraram que o modelo de ensino foi mais eficiente que a abordagem tradicional no que diz respeito ao desenvolvimento de competências técnicas em ES. No entanto, esses resultados não se repetiram para a unidade de Engenharia de Requisitos. A relação desses resultados com trabalhos relacionados e enfoques alternativos, bem como suas limitações, ameaças à validade e implicações para a academia e indústria são descritas nas seções a seguir.

7.2 CONCLUSÕES

Esta pesquisa de doutorado identificou duas lacunas na área de ensino de ES: i) as abordagens de ensino atuais não adotam estratégias que alterem a atual dinâmica de ensino; e ii) existe uma grande dificuldade no desenvolvimento de competências

profissionais dos alunos no ambiente acadêmico. Assim, a principal Questão de Pesquisa (QP) investigada por esta tese foi: "*Se os professores da área de Engenharia de Software (ES) adotarem abordagens de ensino focadas no aluno e práticas de capacitação da indústria, o desenvolvimento de competências técnicas será mais eficiente do que se adotarem abordagens focadas no conteúdo?*".

Nesse sentido, o objetivo dessa pesquisa consistiu na definição de um modelo iterativo baseado nas principais abordagens focadas no aluno que são aplicadas no ensino de Engenharia de Software. Como diferencial, esse modelo adaptou práticas de capacitação adotadas pela indústria de software para o contexto acadêmico a fim de que os estudantes desenvolvam competências técnicas em ES em nível de aplicação.

Buscando responder a QP, inicialmente realizou-se um *survey* com professores e alunos, onde realizaram-se perguntas descritivas e classificatórias relacionadas ao tópicos e abordagens para o ensino de ES. A maioria dos professores respondeu que adota abordagens focadas no professor e 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.

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 6 (seis) unidades de conhecimento mais adotadas no ensino de ES: Engenharia de Requisitos, Processos de Software, Gerenciamento de Projetos, Projetos de Software, Verificação e Validação e Ferramentas e Ambientes. Essas unidades foram o foco da versão inicial do modelo.

Em seguida, realizou-se um mapeamento que buscou responder perguntas relacionais sobre os tópicos de ES do currículo da ACM/IEEE e as práticas específicas do modelo CMMI-DEV. Esse mapeamento serviu de base para definição das perguntas processo-descritas que foram realizadas aos consultores no levantamento sobre práticas da indústria. Assim, foi possível identificar as práticas de *coaching*, *mentoring*, dinâmicas e *workshops* como as mais adotadas na capacitação de profissionais da indústria.

Baseado nos trabalhos relacionados, nas abordagens identificadas no *survey* e nas práticas descritas no levantamento, o modelo de ensino iterativo foi definido a fim de atender ao principal objetivo dessa pesquisa. Esse modelo foi avaliado através de um

painel de especialistas que respondeu perguntas do tipo *design* sobre a documentação e usabilidade. Considerando o parecer dos avaliadores, definiu-se uma nova versão da documentação do modelo, bem como se especificou um sistema *web* que objetivou melhorar a usabilidade e instanciação do mesmo.

Por fim, realizou-se um experimento controlado que buscou responder à seguinte pergunta do tipo causalidade-comparação: "A abordagem humanista é mais efetiva para o desenvolvimento de competências técnicas em ES do que a abordagem tradicional?". Essa pergunta, juntamente com as anteriores, permite responder a QP dessa pesquisa de doutorado. Na primeira turma, onde instanciou-se o modelo para a unidade de conhecimento de Gerenciamento de Projetos, a abordagem humanista se mostrou mais eficiente, havendo uma diferença significativa (P = 0,01016) entre o Tratamento 1 (abordagens focadas no aluno) e Controle (abordagem tradicional). Essa diferença também foi observada entre o Tratamento 2 (abordagens focadas no aluno e práticas da indústria) e Controle, onde o valor de P = 0,00606.

No entanto, esses resultados não se repetiram na segunda turma do experimento (unidade de Engenharia de Requisitos), sendo o valor de P da ANOVA = 0,63130 (não-significativo). Acredita-se que a falta de motivação e de comprometimento dos alunos impactou diretamente os resultados da instanciação do modelo.

Nessa segunda fase do experimento, alguns alunos do grupo de Tratamento 1 responderam que consideram a área de ES "Moderadamente útil", enquanto outros do grupo de Tratamento 2 responderam "Completamente inútil". Todos os alunos do grupo de Controle responderam que a área de ES era "Muito útil" ou "Essencial" para a sua formação. Adicionalmente, na Fase 2 do Experimento, a principal dificuldade relatada pela maioria dos alunos foi a "falta de compromisso por parte da equipe".

Apesar dos resultados do experimento não permitirem validar ou refutar a hipótese alternativa H_0 (Seção 5.3), destaca-se a relevância da proposta para a área de ensino de ES quando esta se propõe a colaborar com o desenvolvimento de competências técnicas em nível de aplicação durante a graduação. A relevância deste trabalho se estende por consequência para a indústria, pois busca atender a demanda de formação profissional na área de ES que, de acordo com um estudo divulgado pela BRASSCOM (2011), deverá chegar a 750 mil profissionais em 2020.

A partir da experiência de uso do modelo no experimento controlado, observou-se que o modelo permite lecionar uma unidade de conhecimento em um curto ciclo iterativo (em média, 9 aulas de 50 minutos). Isso pode vim a impactar diretamente os cursos de graduação da área, principalmente Ciência da Computação, Sistemas de Informação e Engenharia da Computação que, ao contrário do Bacharelado em Engenharia de Software, possui geralmente 3 disciplinas que abordam ES. Desta forma, o professor poderá gerenciar a carga horária disponível de maneira mais adequada e ministrar os tópicos das unidades de conhecimento de forma integrada.

Espera-se que o uso desse modelo possa ajudar a diminuir a insatisfação da indústria com o nível de formação dos profissionais recém-formados que pretendem atuar na área de ES. A adoção do modelo permite que os alunos executem atividades práticas que exploram estímulos sensoriais (visual, auricular, textual e cinestésico) de maneira iterativa através de um ciclo de aprendizagem. Assim, alunos com perfis de aprendizagem diferentes podem, ao avançar em cada etapa do ciclo, conhecer, compreender e aplicar os conteúdos da ES e, consequentemente, desenvolver as competências técnicas das unidades de conhecimento. Como resultado do seguimento desse modelo, a partir das entrevistas realizadas no experimento controlado, baseadas no questionário do SWECOM que são utilizados para seleção de profissionais, observou-se que os alunos aumentaram significativamente o seu nível em relação às competências técnicas requisitadas pela indústria.

7.3 CONTRIBUIÇÕES

As principais contribuições desta pesquisa, obtidas a partir da execução dos métodos empíricos, foram:

1. Um mapeamento entre os currículos de referência da ACM/IEEE (2013) e da SBC (2005) com relação às unidades de conhecimento da área de Engenharia de Software (ver Tabela 3);

2. Uma análise da relevância dos tópicos e da efetividade das abordagens para o ensino de ES, a partir da realização de *surveys* com professores e estudantes (PORTELA, VASCONCELOS e OLIVEIRA, 2015D);

3. Um mapeamento entre os tópicos de ES recomendados pelo currículo de referência da ACM/IEEE (2013) e as práticas específicas do modelo de qualidade CMMI-DEV (SEI, 2010) (Apêndice B);

4. Um levantamento das práticas de capacitação e estratégias de avaliação adotadas em programas de Melhoria de Processo de Software, com consultores que atuam na indústria e também na academia como professores de disciplinas da área de ES (Apêndice C);
5. Um modelo iterativo para o ensino de tópicos de Engenharia de Software, que objetiva desenvolver competências técnicas a partir de abordagens focadas nos alunos e práticas da indústria (Capítulo 4);
6. Uma análise comparativa entre os resultados da aplicação de uma abordagem de ensino tradicional focada no professor e de uma abordagem de ensino iterativa focada no aluno, a partir da condução de um experimento controlado (Apêndice F).

Destaca-se por fim os resultados acadêmicos deste trabalho, como a publicação de diversos artigos em eventos e conferências relevantes para a área de Ensino de Engenharia de Software: i) uma discussão sobre o ensino de Engenharia de Software ser direcionado às necessidades da indústria no *International Workshop on Software Process Education, Training and Professionalism* (IWSPETP) (PORTELA, VASCONCELOS e OLIVEIRA, 2015A); ii) uma discussão sobre como desenvolver competências profissionais durante a graduação na *International Conference on Frontiers in Education: Computer Science and Computer Engineering* (FECS) (PORTELA, VASCONCELOS e OLIVEIRA, 2015B); iii) a proposta de tese de doutorado no Workshop de Teses e Dissertações de Qualidade de Software (WTDQS) (PORTELA, VASCONCELOS e OLIVEIRA, 2015C); iv) os resultados do *survey* no Fórum de Educação em Engenharia de Software (FEES) (PORTELA, VASCONCELOS e OLIVEIRA, 2015D); v) uma proposta de *framework* para o ensino de tópicos de ES no Simpósio Brasileiro de Informática na Educação (SBIE) (PORTELA, VASCONCELOS e OLIVEIRA, 2016A); vi) as etapas e instanciação desse *framework* em uma disciplina de ES no FEES (PORTELA, VASCONCELOS e OLIVEIRA, 2016B); vii) um relato de experiência do uso de práticas da indústria em uma disciplina de ES na *Conference on Software Engineering Education and Training* (CSEE&T) (PORTELA et al., 2017); e viii) a avaliação do modelo por um painel de especialistas no SBIE (PORTELA, VASCONCELOS e OLIVEIRA, 2017).

7.4 LIMITAÇÕES

7.4.1 *Survey* sobre Tópicos e Abordagens

Inicialmente, pretendia-se consultar, além de professores e alunos, profissionais da indústria em relação à relevância dos tópicos de ES. No entanto, não se obteve uma quantidade de respostas considerável. Isso deve-se ao tempo necessário para preenchimento do questionário (32 minutos), conforme relatado por alguns profissionais que foram convidados a responder o *survey*.

Além disto, cerca de 27 professores e 303 alunos foram convidados e não participaram do *survey*, pois não estavam motivados a responder o questionário também devido ao tempo necessário para preenchimento. Essa quantidade de professores e alunos que não respondeu aos questionários (330) foi maior que a amostra obtida (70). Isto impactou diretamente a abrangência e o resultado do *survey*. Esse viés de amostragem foi difícil de tratar, pois o público-alvo reclamou muito da quantidade de tópicos e, consequentemente, do tempo necessário para responder ao questionário. No entanto, nas etapas iniciais da pesquisa, não se poderia restringir a quantidade de tópicos analisados, pois isto poderia comprometer o atendimento do objetivo do *survey*.

A cobertura das abordagens de ensino foi baixa, baseando-se apenas no trabalho de Prikladnicki et al. (2009). No entanto, não é objetivo deste estudo listar todas as abordagens para o ensino de Engenharia de Software, mas sim ter uma ampla cobertura dessas. Por fim, destaca-se que, visando atender ao seu objetivo principal - que consiste em definir um modelo de ensino -, esse estudo considerou apenas a característica de adoção pelos professores para classificar os tópicos mais relevantes e menos relevantes.

7.4.2 Levantamento de Práticas da Indústria

A principal limitação do levantamento de práticas da indústria consistiu na quantidade de participantes. Obteve-se uma baixa amostragem, 10 participantes, considerando que os formulários de levantamento foram divulgados para 40 professores/consultores identificados no site da SOFTEX (*www.softex.br*). Ressalta-se que, a fim de reduzir esse viés da amostragem, realizou-se o levantamento com consultores das regiões norte, nordeste e sudeste do Brasil, que representam cerca de 80% do total da população entrevistada no *survey* (PORTELA, VASCONCELOS e OLIVEIRA, 2015D).

No total, foram obtidas respostas de 5 instituições que implementam MPS. Destaca-se ainda a média de experiência dos consultores/professores entrevistados, sendo de 13 anos e meio de docência na área de Engenharia de Software e 11 anos atuando com consultoria em Melhoria do Processo de Software.

7.4.3 Avaliação por Painel de Especialistas

Dentre as limitações que podem ter influenciado de alguma forma os resultados desta avaliação 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 são doutores que atuam como professores/pesquisadores na área de ensino de ES em diferentes instituições públicas de ensino (UFRPE, UNIFOR e UFLA). Esses professores possuem uma média de 8 anos de experiência docente, sendo que todos já realizaram projetos práticos, discussão de casos práticos, dinâmicas e *workshops* no ensino de ES. Dois desses já adotaram PBL, jogos educativos e *mentoring* em sala de aula e apenas um aplicou g*amification* e *coaching*.

Adicionalmente, ressalta-se que o contato dos especialistas apenas com a documentação do modelo pode ter limitado seu entendimento, levando a avaliações equivocadas. Nesse sentido, utilizou-se uma escala *Likert* nas questões objetivas, permitindo-se a inserção de considerações sobre os valores escolhidos, a fim de complementar a avaliação dos especialistas. Por fim, realizou-se uma reunião de consenso a fim de discutir eventuais equívocos em relação ao entendimento do modelo proposto.

7.4.4 Avaliação por Experimento Controlado

O fato da Fase 2 do Experimento não apontar que uma abordagem humanista possa ser mais efetiva que uma abordagem tradicional não significa que a hipótese nula é verdadeira, apenas que não existe evidência suficiente para confirmá-la. Nesta avaliação de uso do modelo, a eficiência dos tratamentos (abordagens focadas no aluno e práticas de capacitação da indústria) foi observada apenas na Fase 1 do Experimento da unidade de Gerenciamento de Projetos, sendo necessárias algumas observações sobre a sua aplicação na Fase 2 Experimento da unidade de Engenharia de Requisitos.

A condução do experimento foi orientada pelo Princípio IV listado na Seção 4.2, ou seja, todos os grupos realizaram projetos práticos. Assim, todas as equipes tiveram a oportunidade de participar de atividades práticas, a fim de reduzir os problemas de

comparação. Além disso, destaca-se que a decisão prévia sobre quais variáveis seriam ignoradas (a experiência prévia do professor em abordagens humanistas e a motivação dos alunos para estudar ES, por exemplo) e essas podem ser relevantes fora do ambiente de um experimento controlado. Essa redução é característica da postura positivista, adotada pelos pesquisadores deste trabalho e associada ao método experimento controlado (EASTERBROOK et al., 2007).

Outra limitação foi a baixa amostra da população participante do experimento: 2 professores e 30 alunos. Novamente, destaca-se o posicionamento positivista, pois a caracterização de uma ampla população através de técnicas de amostragem requer uma crença no reducionismo (CRESWELL, 2002).

7.5 TRABALHOS FUTUROS

A confiabilidade empírica de uma pesquisa é resultado da aplicação repetida de técnicas por meio de métodos que implicam resultados coerentes (EASTERBROOK et al., 2007). Dessa forma, a confiabilidade é verificada por meio da comparação dos resultados de aplicações repetidas da mesma medida em circunstâncias levemente diferentes. Nesse sentido, pretende-se realizar um experimento controlado de confirmação em outra instituição de ensino, a fim de reforçar as conclusões obtidas nesta pesquisa. Esse experimento já está sendo planejado para o primeiro semestre de 2018 na Universidade Federal Rural da Amazônia (UFRA). Os pesquisadores e professores envolvidos seguirão o protocolo do experimento descrito no Apêndice F deste trabalho.

A fim de tratar a variável “motivação dos alunos”, que impactou diretamente nos resultados do experimento controlado, pretende-se, como trabalho futuro, atualizar o modelo para inserir estratégias de gamificação em todas as suas etapas. A gamificação tem como principais benefícios, reconhecido pelos pesquisadores da área, o engajamento e motivação dos estudantes em atividades de ensino-aprendizagem. Assim, espera-se reduzir o viés relacionado aos alunos que não possuem afinidade com a área de ES.

Adicionalmente, como trabalhos futuros a serem realizados, que estão fora do escopo dessa pesquisa, pretende-se conduzir um *survey* com profissionais da indústria que atuam na área sobre a relevância dos tópicos de ES para a execução de suas atividades. Além disso, pretende-se realizar uma análise mais aprofundada sobre a relevância desses tópicos para a atuação do profissional de software.

Posteriormente, pretende-se realizar um Estudo de Caso que permita observar como ocorre o uso da nova versão do modelo pelos professores e se tal uso tem relação direta com o desenvolvimento de competências técnicas em ES nos alunos. Optou-se por esse método empírico, pois, de acordo com Easterbrook et al. (2007), estudos de caso oferecem compreensão aprofundada de como e por que certos fenômenos ocorrem, e pode revelar os mecanismos pelos quais as relações de causa-efeito acontecem.

Academicamente, pretende-se escrever e publicar artigos sobre a avaliação através do experimento controlado e do próprio estudo de caso. O objetivo é validar os resultados de cada uma dessas etapas 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 modelo atualizado e adequado para o ensino de ES e, consequentemente, eficiente para o desenvolvimento de competências técnicas nos alunos.

REFERÊNCIAS

ABES. **Mercado Brasileiro de Software: panorama e tendências**. 1ª. ed. São Paulo: Associação Brasileira das Empresas de Software, 2017.

ACM/IEEE. **Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science**. New York: ACM, 2013.

ACM/IEEE. **Software Engineering 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering**. Joint Task Force on Computing Curricula. A Volume of the Computing Curricula Series, p. 134. 2014

ANDRADE, A.; JUNIOR, F.; PIMENTEL, J.; BITTENCOURT, J.; SANTANA, T. **Aplicação do Método PBL no Ensino de Engenharia de Software: Visão do Estudante**. ERBASE - Escola Regional de Computação dos Estados da Bahia, Alagoas e Sergipe. 2010.

BEECHAM, S.; HALL, T.; BRITTON, C.; COTTEE, M.; RAINER, A. **Using an Expert Panel to Validate a Requirements Process Improvement Model**. The Journal of Systems and Software, v. 76, 2005.

BESSA, B.; CUNHA, M.; FURTADO, F. **ENGSOFT: Ferramenta para Simulação de Ambientes Reais para auxiliar o Aprendizado Baseado em Problemas (PBL) no Ensino de Engenharia de Software**. Anais do XX Workshop sobre Educação em Informática. Curitiba, 2012.

BLOOM. **Taxonomy of Educational Objectives: The Classification of Educational Goals**. Handbook I, Cognitive Domain: Longmans, 1956.

BOAS, J.; TEIXEIRA, M.; DAMACENO, E.; BRANCHER, J. **GamAPI - Uma API para Gamificação**. INFORMÁTICA NA EDUCAÇÃO: Teoria & Prática, Porto Alegre, v. 20, n. 1, p. 71-80, Jan./Abr. 2017.

BOEHM, B. **Software Engineering**. IEEE Transactions on Computers, Washington, v. 25, n. 12, p. 1226-1241, December 1976.

BRANDÃO, H.; BORGES-ANDRADE, J. **Causas e Efeitos da Expressão de Competências no Trabalho: para entender melhor a noção de Competência**. Revista de Administração Mackenzie, v. 8, n. 3, p. 32-49, 2007.

BRANDÃO, H.; GUIMARÃES, T. **Gestão de Competências e Gestão de Desempenho: tecnologias distintas ou instrumentos de um mesmo constructo?** Revista de Administração de Empresas (FGV), São Paulo, v. 41, n. 1, p. 8-15, 2001.

BRASSCOM. **O Mercado de Profissionais de TI no Brasil**. Associação Brasileira de Empresas de Tecnologia da Informação e Comunicação. São Paulo, p. 22. 2011.

CASPERSEN, M.; CHRISTENSEN, H. **Frameworks in Teaching**. In: BENNEDSEN, J.; CASPERSEN, M.; KÖLLING, M. Reflections on the Teaching of Programming: Springer-Verlag, v. 4821, 2008. Cap. III, p. 190-215.

CASTRO, J.; GIMENES, I.; MALDONADO, J. **Uma Proposta de Plano Pedagógico para a Matéria Engenharia de Software**. Anais do II Curso de Qualidade de Cursos de Graduação da Area de Computação e Informática. Curitiba, 2008. p. 251-270.

CHEN, C-Y; CHONG, P. **Software engineering education: A study on conducting collaborative senior project development**. Journal of Systems and Software, v. 84, n. 3, p. 479-491, 2011.

COUTINHO, E.; BEZERRA, C. **Aplicação de Técnicas e Conceitos no Ensino de Engenharia de Software na Faculdade Lourenço Filho**. Revista Científica da Faculdade Lourenço Filho, Fortaleza, v. 7, n. 1, p. 21-36, 2010.

CRESWELL, J. **Research Design: Qualitative, Quantitative and Mixed Methods Approaches**. 2nd. ed. Sage Publications, 2002.

DETERDING, S.; DIXON, D.; KHALED, R.; NACKE, L. **From Game Design Elements to Gamefulness: Defining "Gamification"**. Proceedings of 15th International Academic MindTrek Conference: Envisioning Future Media Environments, 2011. 9-15.

EASTERBROOK, S.; SINGER, J.; STOREY, M-A.; DAMIAN, D. **Selecting Empirical Methods for Software Engineering Research**. In: SHULL, F.; SINGER, J.; SJøBERG, D. Guide to Advanced Empirical Software Engineering. Springer, 2007. Cap. 11.

ENRICONE, D. **Ser Professor**. 3ª. ed. Porto Alegre: EDIPUCRS, 2002.

FARDO, M. **A Gamificação Aplicada em Ambientes de Aprendizagem**. RENOTE - Revista Novas Tecnologias na Educação, Porto Alegre, v. 11, n. 1, p. 1-9, Dezembro 2013.

FLEMING, N. D.; MILLS, C. **Not Another Inventory, Rather a Catalyst for Reflection**. To Improve the Academy, v. 11, 1992. Cap. 1, p. 137.

FOX, A. **Teaching Practical Software Engineering in the 21st Century**. Invited Talk at Brazilian Conference on Software: Theory and Practice. Belo Horizonte, Brazil. 2015.

FRANÇA, C.; CUNHA, A.; ADJARDE, D.; ALAN, F. **Uma Investigação sobre Estilos de Aprendizagem e Hábitos de Estudo de Engenheiros de Software**. Anais do IX Fórum de Educação em Engenharia de Software. Maringá: CBSoft. 2016. p. 119-129.

GARG, K.; VARMA, V. **Software Engineering Education in India: Issues and Challenges**. Proceedings of 21st Conference on Software Engineering Education and Training, Charleston, 2008. 110-117.

GARY, K. **The Software Enterprise: Practicing Best Practices in Software Engineering Education**. The International Journal of Engineering Education Special Issue on Trends in Software Engineering Education, v. 24, n. 4, p. 705-716, July 2008.

GARY, K.; NAGAPPAN, Y.; VERMA, S.; BRANAGHAN, R. **Assessing evolving conceptual knowledge in software engineering students**. Proceedings of 119th ASEE Annual Conference and Exposition. San Antonio, 2012.

GARY, K.; LINDQUIST, T.; BANSAL, S.; GHAZARIAN, A. **A Project Spine for Software Engineering Curricular Design**. Proceedings of 26th Conference on Software Engineering Education and Training. San Francisco: IEEE. 2013. p. 299-303.

GIMENES, I. **Os Dilemas Didáticos da Engenharia de Software**. Revista da SBC: Engenharia de Software - Qual é o impacto da ES no mercado de Computação e na sociedade como um todo? 1ª. ed. Porto Alegre: SBC, 2015. Cap. 3, p. 21-25.

GNATZ, M. et al. **A Practical Approach of Teaching Software Engineering**. Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03). Madrid: IEEE. 2003. p. 120-128.

GOMES, A.; BARCAUI, A.; SCOFANO, A.; GOMES, D. **Coaching e Mentoring**. 1ª. ed. Rio de Janeiro: Editora FGV, v. I, 2015.

GRILLO, M. **Práticas Docentes e Referenciais Norteadores**. **C**aderno Marista de Educação, Porto Alegre, v. 2, n. 2, p. 41-52, 2003.

HALL, A.; FAGEN, R. **Definition of System**. General Systems: v. 1, 1956. p. 18- 28.

HAZZAN, O.; DUBINSKY, Y. **A Framework for Teaching Software Development Methods**. Proceedings of the 28th international conference on Software engineering (ICSE'06). Shanghai: ACM. 2006. p. 703-706.

HESSE-BIBER, S. **Mixed Methods Research - Mixing Theory and Practice**. The Guilford Press, 2010.

IEEE COMPUTER SOCIETY. **SWECOM - Software Engineering Competency Model**. Version 1.0. IEEE. Washington DC, p. 168. 2014A. (ISBN-10: 0-7695-5373-7).

IEEE COMPUTER SOCIETY. **SWEBOK - Guide to the Software Engineering Body of Knowledge**. Version 3.0. IEEE. Washington DC, p. 335. 2014B. (ISBN-10: 0-7695-5166-1).

JEDLITSCHKA, A.; CIOLKOWSKI, M.; PFAHL, D. **Reporting Experiments in Software Engineering**. In: SHULL, F.; SINGER, J.; SJøBERG, D. Guide to Advanced Topics in Empirical Software Engineering. Springer, 2007. Cap. 8, p. 201-228.

JOYCE, B.; WEIL, M. **Models of Teaching**. 1st. ed. Prentice-Hall, 1972.

JOYCE, B.; WEIL, M.; CALHOUN, E. **Models of Teaching**. 9th. ed. Boston Pearson, 2015.

KITCHENHAM, B.; PFLEEGER, S. **Personal Opinion *Survey*s**. In: Guide to Advanced Empirical Software Engineering. Springer, 2008. Cap. 3, p. 63-92.

KNIBERG, H. **Scrum e XP Direto das Trincheiras**. InfoQ, 2008. Disponivel em: <http://infoq.com/br/minibooks/scrum-xp-from-the-trenches>. Acesso em: Novembro 2015.

KOLB, D. **Experiential Learning: Experience as the Source of Learning and Development**. NJ: Prentice-Hall, 1984.

LETHBRIDGE, T. **What Knowledge Is Important to a Software Professional?** IEEE Computer Society, Ottawa, May 2000. 44-50.

LETHBRIDGE, T.; DIAZ-HERRERA, J.; LEBLANC, R. THOMPSON, J. **Improving software practice through education: Challenges and future trends**. Proceedings of the Conference Future of Software Engineering, 2007 (FOSE '07). Minneapolis: IEEE. 2007. p. 12-28.

MAGALHÃES, S.; WANDERLEY, M.; ROCHA, J. Desenvolvimento de Competências: O Futuro Agora! **Revista Treinamento & Desenvolvimento**, São Paulo, p. 12-14, Janeiro 1997.

MALIK, B.; ZAFAR, S. **A Systematic Mapping Study on Software Engineering Education**. International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering, v. 6, n. 11, p. 3343-3353, 2012.

MARQUES, M.; QUISPE, A.; OCHOA, S. **A Systematic Mapping Study on Practical Approaches to Teaching Software Engineering**. Frontiers in Education Conference. Madrid: IEEE. 2014. p. 1-8.

MEAD, N. **Software engineering education: How far we've come and how far we have to go.** Journal of Systems and Software, v. 82, n. 4, p. 571-575, 2009.

MEC. **Diretrizes Curriculares de Cursos da Área de Computação**. Departamento de Políticas do Ensino Superior. Brasília, p. 23. 2012.

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? 1ª. ed. Porto Alegre: Sociedade Brasileira de Computação, 2015. Cap. 1, p. 11-15.

MIZUKAMI, M. **Ensino: As Abordagens do Processo**. São Paulo: Epu, 1986.

MONSALVE, E.; WERNECK, V.; LEITE, J. **SimulES-W: Um Jogo para o Ensino de Engenharia de Software**. Anais do III Fórum de Ensino de Engenharia de Software (FEES). Salvador, 2010.

MORENO, A; SANCHEZ-SEGURA, M-I; MEDINA-DOMINGUEZ, F; CARVAJAL, L. **Balancing software engineering education and industrial needs**. Journal of Systems and Software, v. 85, n. 7, p. 1607-1620, 2012.

MULLER, D. **This Will Revolutionize Education**. Youtube, 01 dez. 2014. Disponível em <https://www.youtube.com/watch?v=GEmuEWjHr5c >. Acesso em: Outubro 2017.

NUNES, D. **Competências do Engenheiro de Software**. Revista da SBC: Engenharia de Software - Qual é o impacto da ES no mercado de Computação e na sociedade como um todo? 1ª. ed. Porto Alegre: Sociedade Brasileira de Computação, 2015. Cap. 2, p. 16-20.

NUNES, D.; REIS, C.; REIS, R. **Educação em Engenharia de Software**. In: ALMEIDA, E. S.; MASIERO, P. C. A carreira do pesquisador em Engenharia de Software: princípios, conceitos e direções. 1ª. ed. Salvador: UFBA, 2010. Cap. 5, p. 132-181.

NUNES, D. J.; YAMAGUTI, M. H.; NUNES, I. **Refinamento de Competências do Egresso do Curso de Engenharia de Software**. Anais do IX Fórum de Educação em Engenharia de Software. Maringá: CBSoft. 2016. p. 143-146.

OHLSSON, L.; JOHANSSON, C. **A Practice Driven Approach to Software Engineering Education.** IEEE Transactions On Education, August 1995. 291-295.

PASSI, B. K.; SINGH, L. C.; SANSANWAL, D. N. **Models of Teaching: report of the three-phase study of CAM and ITM**. 1st. ed. New Delhi: National Council of Education Research and Training, 1991.

PATELIYA, Y. P. **An Introduction to Modern Models of Teaching**. International Journal for Research in Education, v. 2, n. 2, p. 125-129, February 2013. ISSN ISSN:2320-091X.

PAUL, A. **Stop Propagating the Learning Styles Myth**. Computers & Education, v. 106, p. 166-171, March 2017. ISSN 0360-1315.

PORTELA, C.; VASCONCELOS, A.; OLIVEIRA, S. **An Approach to Teaching Software Process Oriented to Quality Models**. Proceedings of International Workshop Software Process Education, Training and Professionalism. Gothenburg: SPICE Conference. 2015A. p. 70-74.

PORTELA, C.; VASCONCELOS, A.; OLIVEIRA, S. **How to Develop Competencies and Abilities Professional in Software Engineering in Undergraduate Students?** Proceedings of International Conference on Frontiers in Education: Computer Science and Computer Engineering. Las Vegas: WORLDCOMP 2015. 2015B. p. 91-94.

PORTELA, C.; VASCONCELOS, A.; OLIVEIRA, S. **Uma Abordagem para o Ensino de Engenharia de Software Baseada em Estratégias de Capacitação de Programas de Melhoria do Processo de Software**. Anais do Workshop de Teses e Dissertações de Qualidade de Software. Manaus: SBQS. 2015C.

PORTELA, C.; VASCONCELOS, A.; OLIVEIRA, S. **Análise da Relevância dos Tópicos e da Efetividade das Abordagens para o Ensino de Engenharia de Software**. Anais do VIII Fórum de Educação em Engenharia de Software. Belo Horizonte: SBC. 2015D. p. 24-35.

PORTELA, C.; VASCONCELOS, A.; OLIVEIRA, S. **FRAMES: Uma Proposta de Framework para o Ensino de Tópicos da Engenharia de Software**. Anais do XXVII Simpósio Brasileiro de Informática na Educação. Uberlândia: SBC. 2016A. p. 1361-1365.

PORTELA, C.; VASCONCELOS, A.; OLIVEIRA, S. **FRAMES: Um Framework para o Ensino-Aprendizagem dos Tópicos de Engenharia de Software dos Currículos de Referência da ACM/IEEE e SBC**. Anais do IX Fórum de Educação em Engenharia de Software. Maringá: SBC. 2016B. p. 41-52.

PORTELA, C.; VASCONCELOS, A.; OLIVEIRA, S; SOUZA, M. **The Use of Industry Training Strategies in a Software Engineering Course: An Experience Report**. Proceedings of 30th Conference on Software Engineering Education and Training (CSEE&T). Savannah: IEEE. 2017. p. 29-36.

PORTELA, C.; VASCONCELOS, A.; OLIVEIRA, S. **Um Modelo Iterativo para o Ensino de Engenharia de Software Baseado em Abordagens Focadas no Aluno**. Anais do VIII Simpósio Brasileiro de Informática na Educação. Recife: SBC. 2017. p. 304-313.

PRIKLADNICKI, R. ALBUQUERQUE, A.; WANGENHEIM, C.; CABRAL, R. **Ensino de Engenharia de Software: Desafios, Estratégias de Ensino e Lições Aprendidas**. Anais do II Fórum de Educação em Engenharia de Software. Fortaleza, 2009.

ROCHA, A. R. **GT/T4: Melhorias na Capacitação de Pessoas. Painel Melhorias no Programa MPS.BR e no Modelo MPS**. Blumenau: Anais do XIII Simpósio Brasileiro de Qualidade de Software (SBQS). 2014. p. 9.

RODRIGUES, N.; ESTRELA, N. **Simple Way: Ensino e Aprendizagem de Engenharia de Software Aplicada através de Ambiente e Projetos Reais**. Anais do VIII Simpósio Brasileiro de Sistemas de Informação. São Paulo, 2012. p. 722-733.

SANTOS, R. **Abordagens do Processo de Ensino e Aprendizagem**. Revista Integração, Rio Grande do Sul, v. XI, n. 40, p. 19-31, Jan/Fev/Mai 2005.

SANTOS, R.; MAGALHÃES, C.; CORREIA-NETO, J.; SOUZA, E.; VILAR G. **Ferramentas, Métodos e Experiências no Ensino de Engenharia de Software: um Mapeamento Sistemático**. Anais do III Congresso Brasileiro de Informática na Educação. 2014. p. 544-548.

SANTOS, S.; BATISTA, M.; CAVALCANTI, A.; ALBUQUERQUE, J.; MEIRA, S. **Applying PBL in Software Engineering Education**. Proceedings of 22nd Conference on Software Engineering Education and Training (CSEE&T). Hyderabad, India. 2009. p. 182-189.

SANTOS, S.; FIGUERÊDO, C.; WANDERLEY, F. **PBL-Test: a Model to Evaluate the Maturity of Teaching Processes in a PBL Approach**. Proceedings of Frontiers in Education Conference (FIE). Oklahoma, EUA. 2013. p. 595-601.

SANTOS, S.; ALEXANDRE, G.; RODRIGUES, A. **Applying PBL in Project Management Education: a Case Study of an Undergraduate Course**. Proceedings of Frontiers in Education Conference (FIE). El Paso, EUA. 2015. p. 1-8.

SAVI, R. **Avaliação de Jogos voltados para a Disseminação do Conhecimento**. Tese de Doutorado. Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento: Universidade Federal de Santa Catarina. 2011. p. 236.

SBC. **Currículo de Referência da SBC para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia de Computação**. Porto Alegre: Sociedade Brasileira da Computação, 2005.

SEI. Software Engineering Institute. **Capability Maturity Model Integration for Development (CMMI-DEV V1.3)**, 2010. Disponivel em: <http://www.sei.cmu.edu/reports/10tr033.pdf>. Acesso em: Agosto 2015.

SHAW, M. **Software Engineering Education:** A Roadmap. Proceedings of the 22nd International Conference on Software Engineering. ACM Press. 2000. p. 371-380.

SHKOUKANI, M. Proposed Model to Find the Gap Between Academic Supply and Industry Demand in Software Engineering Field in Jordan. **International Journal of Advanced Computational Engineering and Networking**, v. 1, n. 2, p. 2320-2106, April 2013.

SOARES, M. **Uma Experiência de Ensino de Engenharia de Software Orientada a Trabalhos Práticos**. Anais do I Fórum de Educação em Engenharia de Software. Campinas, 2008.

SOFTEX. Associação para Promoção da Excelência do Software Brasileiro. **Guia Geral MPS de Software**, 2012. Disponivel em: <http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Software_2012.pdf>. Acesso em: Agosto 2015.

SOWE, S.; STAMELOS, I.; DELIGIANNIS, I. **A Framework for Teaching Software Testing using F/OSS Methodology**. In: Open Source Systems. Como: Springer, v. 203, 2006. Cap. 6, p. 261-266.

STEWART, K. **The Mediating Role of Classroom Social Environment between Teacher Self-efficacy and Student Adjustment**. University of South Florida. Graduate Theses and Dissertations, p. 154. 2014.

THIRY, M.; ZOUCAS, A.; GONÇALVES, R. **Promovendo a Aprendizagem de Engenharia de Requisitos de Software através de um Jogo Educativo**. Anais do XXI Simpósio Brasileiro de Informática na Educação. João Pessoa: 2010.

TRAVASSOS, G.; GUROV, D.; AMARAL, E. **Introdução à Engenharia de Software Experimental**. COPPE/UFRJ, Rio de Janeiro, Relatório técnico: RT-ES- 590/02, 2002. Disponível em <http://www.ufpa.br/cdesouza/teaching/topes/4-ES-Experimental.pdf>.

WANGENHEIM, C.; SILVA, D. **Qual Conhecimento de Engenharia de Software é Importante para um Profissional de Software?** Anais do II Fórum de Educação em Engenharia de Software. Fortaleza, 2009.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M.; REGNELL, B.; WESSLÉN, A. **Experimentation in Software Engineering**. 1. ed. Springer, 2012.

ZABALA, A.; ARNAU, L. **Como Aprender e Ensinar Competências**. Ponta Grossa: Artmed, 2010.

ZICHERMANN, G.; CUNNINGHAM, C. **Gamification by Design: Implementing Game Mechanics in Web and Mobile Apps**. O'Reilly Media; 1 ed., 2011.

APÊNDICE A – RELATÓRIO DE PESQUISA DO *SURVEY*

1. INTRODUÇÃO

Atualmente, as matrizes curriculares de disciplinas da computação são definidas com base em currículos de referência. No entanto, acreditamos que seria mais interessante consultar os envolvidos no processo de ensino/aprendizagem de tópicos de Engenharia de Software.

Alguns *surveys* já realizados na área de computação constataram que determinados tópicos são enfatizados nos currículos das disciplinas, mas que na atuação profissional estes não são necessários para a execução das atividades. Por outro lado, existem tópicos que são de extrema relevância para que o aluno desenvolva habilidades técnicas, pessoais e de comunicação.

Este *survey* incorpora questionamentos e a base dos protocolos de pesquisas realizadas na área de ensino de computação, a fim de comparar seus resultados. No entanto, diferente destes, há um enfoque na disciplina de Engenharia de Software.

Os resultados deste *survey* podem ser úteis para definição de ementas de Engenharia de Software de cursos de graduação e de estruturas curriculares de departamentos de treinamento organizacional. Estudantes e professores podem tomar os resultados deste estudo como base a fim de continuar e contribuir na identificação de quais tópicos são relevantes para a formação profissional na área de Engenharia de Software.

2. O SURVEY

Os participantes do *survey* foram recrutados através de abordagem direta e por divulgação via internet. Os questionários do *survey* estavam disponíveis via formulário *web*, onde foi utilizada a plataforma do *Google Forms* que permite publicar e armazenar dados em uma planilha. Os participantes indicaram que o tempo estimado para responder todas as questões foi em média 32 minutos.

Estes participantes foram questionados a respeito de 83 tópicos de Engenharia de Software identificados nos currículos de referência da ACM/IEEE e SBC. Estes currículos são adotados por vários cursos de Ciência da Computação e Sistemas de Informação de universidades públicas e privadas do Brasil. Nós também identificamos, a partir de uma revisão da literatura, as principais abordagens de ensino e métodos de avaliação adotados em disciplinas de Engenharia de Software. Não é nossa pretensão listar todos os tópicos e abordagens de ensino de Engenharia de Software, mas sim ter uma ampla cobertura destes.

Nós realizamos pré-testes de aplicação, a fim de revisar e validar as questões do *survey*, com 4 (quatro) alunos e 2 (dois) professores que pesquisam e atuam na área de Engenharia de Software. A fim de reduzir o viés de amostragem, nós aplicamos o *survey* em todas regiões do Brasil: Norte, Centro-Oeste, Nordeste, Sudeste e Sul. O viés de

amostragem causa problemas em generalizar os resultados da pesquisa, pois os entrevistados podem não ser uma amostra representativa da população-alvo. Desta forma, obtivemos respostas de 20 instituições de ensino públicas e privadas.

A instituição que teve o maior número de participantes foi a Universidade Federal do Pará, da qual participaram 3 professores e 20 estudantes. Quanto à região, 10 instituições do Nordeste participaram.

2.1. Questões de Pesquisa

A Tabela 1 mostra as 3 (três) questões feitas para os professores da disciplina de Engenharia de Software a fim de identificar quais tópicos e abordagens de ensino estão sendo adotados.

Tabela 1 - Questões do Survey para os Professores

Questão	**Opções de Respostas**
Q1. *Quais currículos de referências são adotados na definição da ementa da disciplina de Engenharia de Software?*	( ) *Curriculum Guidelines* da ACM/IEEE; ( ) Currículo de Referência da SBC; ( ) Outros; ( ) Nenhum.
Q2. *Quais tópicos de Engenharia de Software estão sendo contemplados na ementa destas disciplinas?*	Para cada um dos 83 tópicos de ES o professor deveria responder: ( ) Contemplado ( ) Não Contemplado
Q3. *Quais abordagens de ensino são adotadas na disciplina de Engenharia de Software?*	A. Métodos de Ensino (quanto ao papel do professor, objetivos, dentre outros.); B. Abordagens de Ensino (aulas expositivas, uso de jogos, dentre outros.); C. Estratégias de Avaliação (provas individuais, trabalhos práticos, projeto de software, outros.).

A Tabela 2 mostra as 3 (três) perguntas feitas para os alunos concluintes da disciplina de Engenharia de Software a fim de analisar a aprendizagem e sua preferência por abordagens de ensino. Para as questões Q1 e Q2 foi utilizada uma escala *Likert* de 0 até 5.

Tabela 2 - Questões do *Survey* para os Alunos

Questão	**Opções de Respostas**
Q1. *O quão útil considera a disciplina de Engenharia de Software para a sua formação profissional?*	0. completamente inútil 1. quase nunca útil 2. ocasionalmente útil 3. moderadamente útil 4. muito útil 5. essencial
Q2. *O quanto aprendeu das Unidades de Conhecimento ministradas durante a disciplina de Engenharia de Software?*	Para cada uma das 125 aprendizagens de ES o aluno deveria responder: 0. Não aprendi absolutamente nada 1. Aprendi vagamente 2. Aprendi o básico

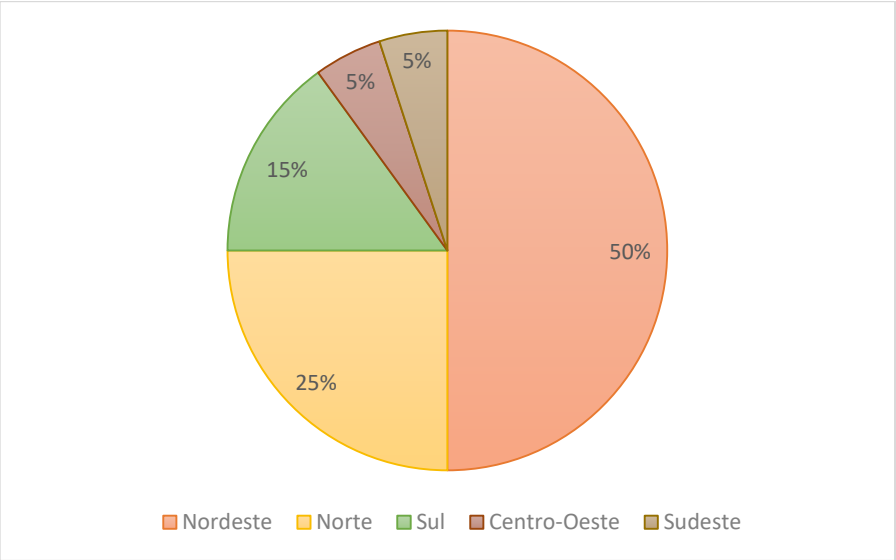
Questão	**Opções de Respostas**
	3. Aprendi moderadamente 4. Aprendi muito 5. Aprendi em profundidade
Q3. *Quais abordagens de ensino de Engenharia de Software considera melhor para sua aprendizagem?*	A. Métodos de Ensino (quanto ao papel do professor, objetivos, dentre outros.); B. Abordagens de Ensino (aulas expositivas, uso de jogos, dentre outros.); C. Estratégias de Avaliação (provas individuais, trabalhos práticos, projeto de software, outros.).

2.2. Os Participantes

Nós recebemos respostas de 70 participantes de uma ampla variedade de instituições. A amostra buscou balancear as instituições de diversas regiões do Brasil, a fim de tratar o viés de uma baixa amostra. O *survey* foi divulgado para mais de 50 professores e mais de 350 alunos. No entanto, muitos destes não responderam (segundo relatos) devido ao tempo estimado para análise e preenchimento. Obtivemos resposta de 23 professores e 47 alunos. Esse viés de amostragem foi difícil de tratar, pois os respondentes reclamaram muito da quantidade de tópicos e, consequentemente, do tempo necessário para responder ao questionário. Neste primeiro momento, não poderíamos restringir a quantidade de tópicos analisados, pois isto comprometeria o objetivo da pesquisa. A partir dos resultados deste *survey*, poderemos restringir a quantidade de tópicos sob análise a partir da identificação de quais são mais relevantes.

Apesar da baixa amostra, os participantes representam 12 estados do Brasil, cerca de 44% do total de estados. Destes, 50% são de instituições da região Nordeste, 25% do Norte, 15% do Sul, 5% do Centro-Oeste e 5% do Sudeste, conforme sintetiza o Gráfico 1. A maioria dos participantes, 80%, são de instituições públicas de ensino e 20% são de instituições privadas.

Gráfico 1 - Percentual de Instituições Participantes por Região



A média de idade dos professores é de 38 anos. Em média, o ano da última formação destes professores foi 2010, sendo a formação mais recente em 2015 e a mais antiga em 1998, há 17 anos. Quanto à formação, 4% possui Especialização e a maioria, 39%, possuem Mestrado ou Doutorado. 19% dos professores entrevistados possuem pós-doutorado. Em média, estes professores lecionam a disciplina de Engenharia de Software há 8 anos, sendo o maior tempo de atuação 25 anos e o menor 1 ano.

A média de idade dos alunos é de 24 anos. Em média, o ano de conclusão da disciplina foi 2013, sendo a formação mais recente em 2015 e a mais antiga em 2005, há 10 anos.

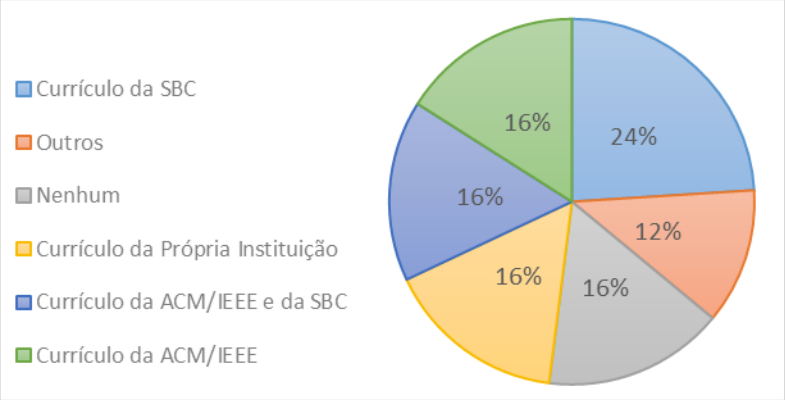
3. OS RESULTADOS DO *SURVEY*

3.1. Professores e a Relevância dos Tópicos de Engenharia de Software

A) Currículos de Referências adotados na ementa de Engenharia de Software

Em relação aos currículos de referência adotados pelos professores, os resultados são apresentados no Gráfico 2.

Gráfico 2 - Currículos de Referência adotados pelos Professores



Observa-se que a maioria (24%) dos professores adota o currículo de referência da SBC. Há um equilíbrio quanto as demais abordagens, pois 16% utiliza o currículo da ACM/IEEE e outros 16% adota estes 2 currículos. 16% dos professores adota um currículo definido pela própria instituição e 12% considera outras abordagens, como SWEBOK e ENADE. Por fim, 16% dos professores não adota nenhum currículo de referência na definição de suas ementas.

B) Tópicos contemplados na ementa da disciplina de Engenharia de Software

Apresentamos os dados em dois formatos complementares: como listas categorizadas de tópicos e como gráficos, retratando os 33 tópicos mais relevantes, de acordo com a adoção destes pelos professores, e os 31 menos relevantes. Para se chegar a estes números, consideram-se os tópicos mais relevantes aqueles que são adotados por mais de 70% dos professores entrevistados e os menos relevantes aqueles que são adotados por menos de 40% dos professores. A Tabela 1 mostra informações sobre a relevância de cada tópico. Nós dividimos os tópicos no *survey* em unidades de

conhecimento para facilitar a localização dos tópicos e identificar conjuntos de tópicos quando as repostas forem similares. Na Tabela 1, o círculo verde identifica que o tópico é o mais relevante para uma unidade de conhecimento, enquanto o círculo vermelho significa que o tópico é considerado menos relevante para um conjunto, de acordo com os professores participantes do *survey*. Da mesma forma, o triângulo verde significa que o tópico é um dos 33 mais relevantes e o triângulo invertido com a cor vermelho significa que é um dos 31 menos relevantes.

Considerando a quantidade dos tópicos relevantes apresentados na Tabela 3, aqueles que são adotados por mais de 70% dos professores, observa-se que 33% destes tópicos são da área de Engenharia de Requisitos, 18% são da área de Processos de Software e as áreas de Projetos de Software e Gerenciamento de Projetos de Software possui 15% dos tópicos relevantes. A área de Verificação e Validação de Software possui 12% e a área de Ferramentas e Ambientes possui 6%. Estes dados estão sintetizados no Gráfico 3.

Gráfico 3 - Percentual de Tópicos Relevantes da Engenharia de Software

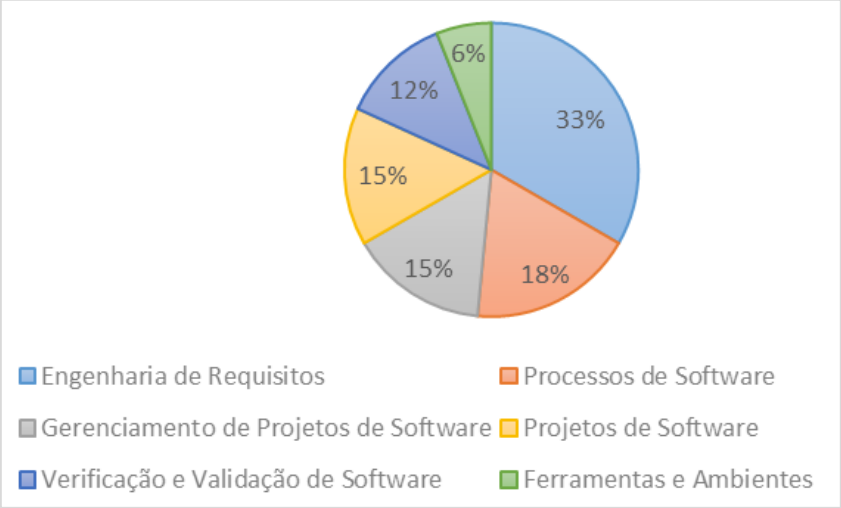


Tabela 3 - Tópicos da Área de Engenharia de Software (ACM/IEEE e SBC)

Unidade de Conhecimento	*Tópicos Abordados*	*Relevância*	*Ranking Unidade*	*Ranking Área*
Processos de Software	*1. Interação do software com o seu ambiente específico*	62%		
	2. Introdução aos modelos de processo de software (por exemplo, cascata, incremental, ágil)	100%	●	▲
	3. Programação em larga escala vs. programação individual	38%	●	▼
	4. Avaliação de modelo de processo de software	81%		▲
	5. Conceitos de qualidade de software	95%		▲
	6. Melhoria de processo	71%		▲
	7. Modelo de maturidade e capacidade de processo de software	71%		▲
	8. Medição de processo de software	81%		▲
Gerenciamento de Projetos de Software	*1. Participação da equipe (responsabilidades, reuniões, agenda de trabalho, resolução de conflitos, etc)*	81%	●	▲
	2. Estimativa de esforço	76%		▲
	3. Riscos (segurança, mercado, financeiro, tecnológico, pessoas, qualidade, estrutura e processo)	67%		
	4. Gerenciamento de equipe (organização e tomada de decisão, identificação de papel e avaliação de desempenho)	62%		
	5. Gerenciamento de Projeto (planejamento e acompanhamento, ferramentas de gerenciamento de projeto e análise de custo/benefício)	76%		▲
	6. Técnicas de medição e estimativa de software	76%		▲
	7. Garantia de qualidade de software e o papel das medições	67%		
	8. Riscos (identificação e gestão, análise e avaliação, tolerância e planejamento)	81%	●	▲
	9. Riscos associados a ferramentas	10%	●	▼

Unidade de Conhecimento	*Tópicos Abordados*	*Relevância*	*Ranking Unidade*	*Ranking Área*
Ferramentas e Ambientes	*1. Gerência de configuração e controle de versão*	71%		▲
	2. Gerenciamento de release	29%		▼
	3. Análise de requisitos e ferramentas de modelagem	100%	●	▲
	4. Ferramentas de testes (análise estática e dinâmica)	43%		
	5. Ambientes de programação que automatizam partes dos processos de desenvolvimento (integração contínua)	29%		▼
	6. Conceitos e mecanismos de integração da ferramenta	24%	●	▼
Engenharia de Requisitos	*1. Descrição dos requisitos funcionais usando, por exemplo, casos de uso ou histórias do usuário*	100%	●	▲
	2. Propriedades de requisitos, incluindo a consistência, validade, integridade e viabilidade	86%		▲
	3. Elicitação de requisitos de software	95%		▲
	4. Descrição dos dados do sistema utilizando, por exemplo, diagramas de classe ou diagramas entidade-relacionamento	100%	●	▲
	5. Requisitos não-funcionais e sua relação com a qualidade de software	100%	●	▲
	6. Avaliação e utilização de especificações de requisitos	90%		▲
	7. Análise de requisitos (técnicas de modelagem)	95%		▲
	8. Aceitabilidade da certeza/incerteza (considerações em relação ao comportamento do software/sistema)	43%	●	
	9. Prototipagem	90%		▲
	10. Conceitos básicos de especificação de requisitos formais	57%		
	11. Especificação de requisitos	100%	●	▲
	12. Validação de requisitos	95%		▲
	13. Rastreamento de requisitos	71%		▲

Unidade de Conhecimento	*Tópicos Abordados*	*Relevância*	*Ranking Unidade*	*Ranking Área*
Projetos de Software	*1. Princípios de projeto de sistema: níveis de abstração (projeto arquitetônico e projeto detalhado), separação de interesses, a ocultação de informações, o acoplamento e coesão, reuso de estruturas padrão*	86%	●	▲
	2. Paradigmas de projeto, tais como projeto estruturado, orientada a objetos, orientado a evento, a nível de componente, orientada a aspecto, orientada a função, orientada a serviços	71%		▲
	3. Modelos estruturais e comportamentais de projetos de software	71%		▲
	4. Padrões de projeto	71%		▲
	5. Relações entre requisitos e projetos: a transformação de modelos, elaboração de contratos	52%		
	6. Conceitos de arquitetura de software e padrões arquiteturais (por exemplo, cliente-servidor)	81%		▲
	7. Projetos de refatoração utilizando padrões de projeto	38%		▼
	8. Uso de componentes no projeto (por exemplo, a construção de uma interface gráfica usando um conjunto de widgets padrão)	43%		
	9. Qualidades internas do projeto (eficiência e desempenho, redundância e tolerância a falhas, rastreabilidade de requisitos)	38%		▼
	10. Qualidades externas do projeto (funcionalidade, confiabilidade, desempenho e eficiência, facilidade de utilização, manutenção, portabilidade)	57%		
	11. Medição e análise da qualidade do projeto	48%		
	12. Estruturas de aplicativos	29%		▼
	13. Middleware	10%	●	▼
	14. Princípios de projeto seguro e codificação	19%		▼

Unidade de Conhecimento	*Tópicos Abordados*	*Relevância*	*Ranking Unidade*	*Ranking Área*
Construção de Software	*1. Práticas de codificação: técnicas, expressões idiomáticas/padrões, mecanismos para a construção de programas de qualidade*	33%		▼
	2. Padrões de Codificação	48%	●	
	3. Estratégias de integração	29%		▼
	4. Contexto de desenvolvimento (análise de impacto de mudanças, atualização)	33%		▼
	5. Potenciais problemas de segurança em programas	24%	●	▼
Verificação e Validação de Software	*1. Conceitos de verificação e validação*	95%	●	▲
	2. Inspeções, revisões, auditorias	76%		▲
	3. Tipos de testes, incluindo interface homem-computador, usabilidade, confiabilidade, segurança, conformidade com especificação	90%		▲
	4. Fundamentos de teste (unidade, integração, validação e testes de sistema, criação do plano de teste e casos de teste, caixa-preta e caixa branca, teste de regressão, automação de teste, controle de defeitos)	86%		▲
	5. Limitações do teste em domínios específicos, tais como sistemas paralelos ou de segurança crítica	33%		▼
	6. Abordagens estáticas e abordagens dinâmicas para verificação	38%		▼
	7. Desenvolvimento orientado a testes	38%		▼
	8. Planejamento de Validação; documentação para validação	43%		
	9. Testes orientada a objetos; teste de sistemas	57%		
	10. Verificação e validação de artefatos não-código (documentação, manuais, materiais de treinamento)	48%		
	11. Registo de falhas, detecção de falhas e suporte técnico para tais atividades	38%		▼
	12. Estimativa de falhas	14%	●	▼

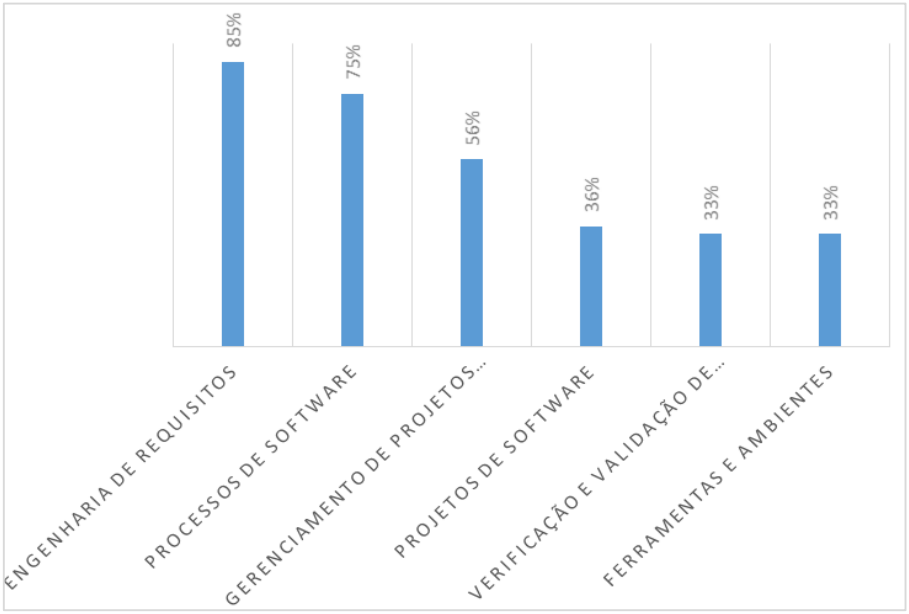
Unidade de Conhecimento	*Tópicos Abordados*	*Relevância*	*Ranking Unidade*	*Ranking Área*
Evolução de Software	*1. O desenvolvimento de software no contexto de grandes bases de código pré-existentes (mudança no Software, localização, refatoração)*	33%	🔴	🔻
	2. Evolução do Software	52%		
	3. Características de manutenibilidade	67%	🟢	
	4. Reengenharia de sistemas	62%		
	5. Reuso de Software (segmentos de código, bibliotecas e frameworks, componentes e linhas de produto)	67%	🟢	
Confiabilidade de Software	*1. Conceitos de engenharia de confiabilidade de software*	29%		🔻
	2. Confiabilidade de software, confiabilidade do sistema e comportamento perante falha	38%	🟢	🔻
	3. Conceitos e técnicas de ciclo de vida de falha	19%		🔻
	4. Modelos de confiabilidade software	14%		🔻
	5. Técnicas de tolerância a falhas de software e modelos	19%		🔻
	6. Práticas de engenharia de confiabilidade software	10%	🔴	🔻
	7. Análise baseada na medição de confiabilidade de software	10%	🔴	🔻
Métodos Formais	*1. Papel das técnicas formais de especificação e análise do ciclo de desenvolvimento de software*	29%	🟢	🔻
	2. Linguagens de declaração e abordagens de análise (incluindo idiomas para a escrita e análise de pré e pós-condições, como a OCL, JML)	19%		🔻
	3. Abordagens formais para modelagem de software e análise	19%		🔻
	4. Ferramentas de apoio a métodos formais	10%	🔴	🔻

Legenda: Tópico Mais Relevante da Unidade de Conhecimento (🟢) Tópico Menos Relevante da Unidade de Conhecimento (🔴)

Tópicos Mais Relevantes da Engenharia de Software (🔺) Tópicos Menos Relevantes da Engenharia de Software (🔻)

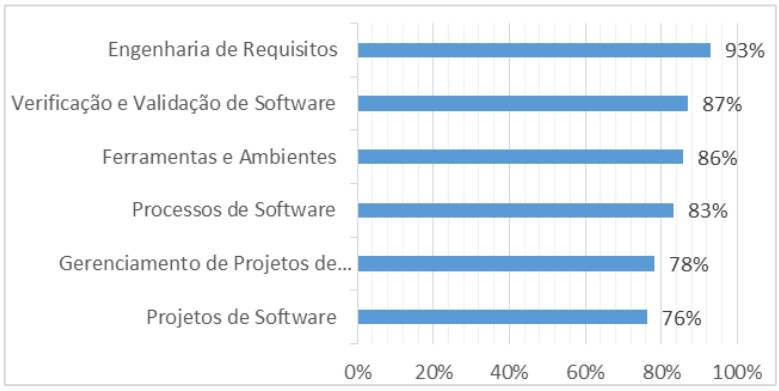
Também é possível analisar o percentual de tópicos relevantes para cada unidade de conhecimento. No Gráfico 4, observa-se que a Engenharia de Requisitos possui a maior quantidade de tópicos relevantes, ou seja, 85% de seus tópicos são adotados por mais de 70% dos professores.

Gráfico 4 - Percentual de Tópicos Relevantes por Área



A partir da análise do percentual de tópicos relevantes, é possível analisar a média de relevância obtida pelos tópicos de cada unidade de conhecimento. No Gráfico 5, podemos confirmar que a área de Engenharia de Requisitos é considerada a área de maior relevância (a média de relevância de seus tópicos é de 93%).

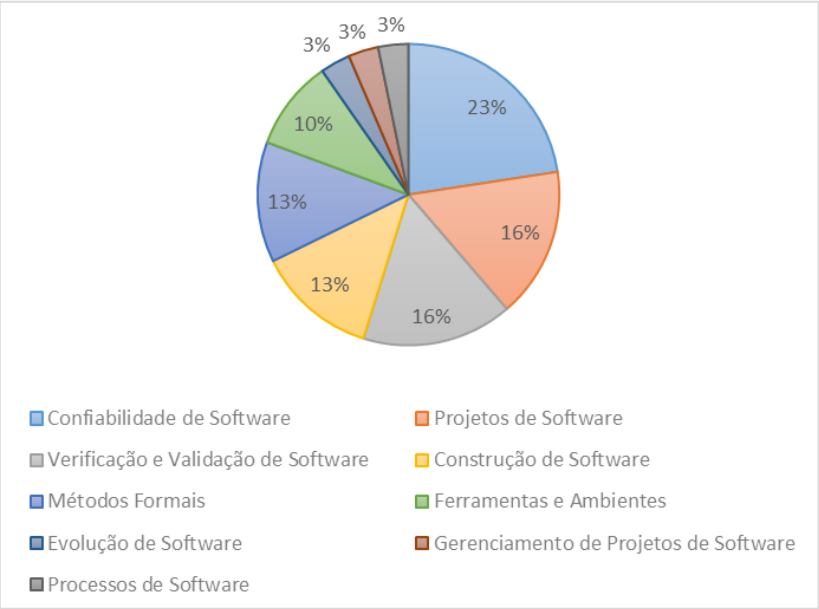
Gráfico 5 - Média de Tópicos Relevantes por Unidade de conhecimento



Analisando a Tabela 3, considerando a quantidade dos tópicos menos relevantes, aqueles que são adotados por menos de 40% dos professores, observa-se que 23% destes tópicos são da área de Confiabilidade de Software, 16% são das áreas de Projetos de Software e Verificação e Validação de Software. As áreas de Construção de Software e Métodos Formais possuem 13% e a área de Ferramentas e Ambientes possui 10% de tópicos menos relevantes. Por fim, as áreas de Evolução de Software, Gerenciamento de

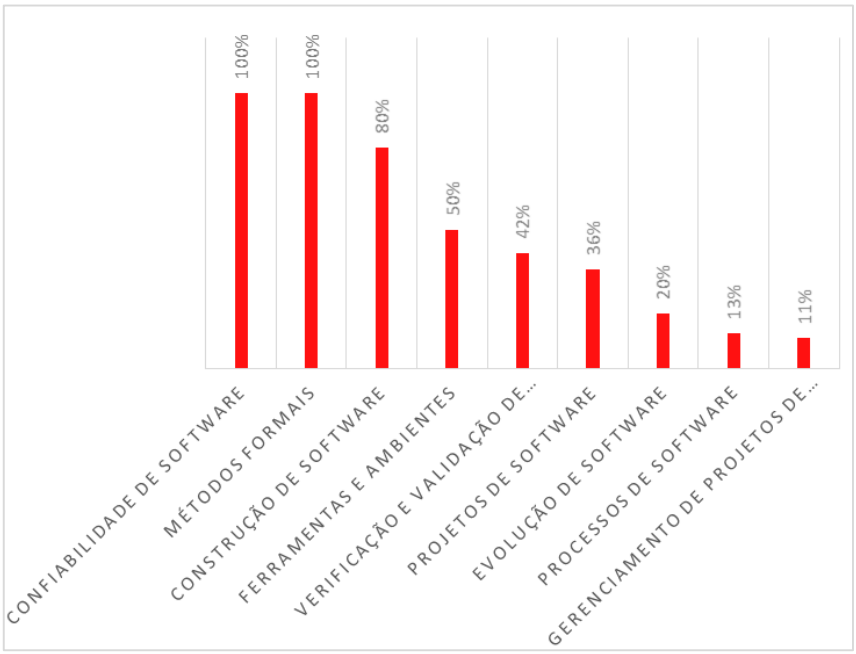
Projetos de Software e Processos possuem apenas 3% dos tópicos menos relevantes. Estes percentuais estão sintetizados no Gráfico 6.

Gráfico 6 - Percentual de Tópicos Menos Relevantes da Engenharia de Software



O percentual de tópicos menos relevantes para cada unidade de conhecimento é apresentado no Gráfico 7, onde observa-se que as áreas de Confiabilidade de Software e Métodos Formais possuem a maior quantidade de tópicos, ou seja, 100% de seus tópicos são adotados por menos de 40% dos professores.

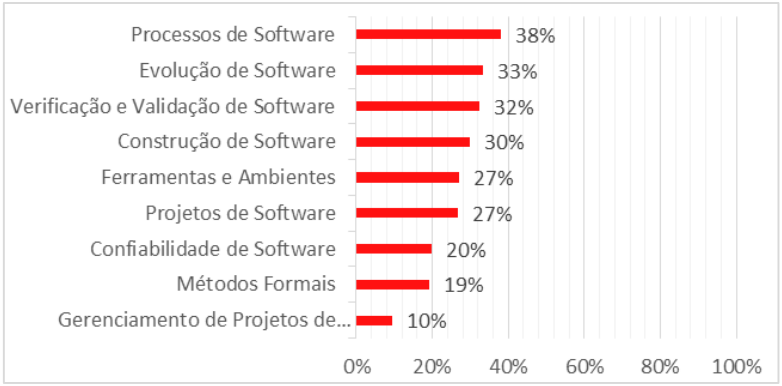
Gráfico 7 - Percentual de Tópicos Menos Relevantes por Área



Quanto a média de relevância obtida pelos tópicos destas áreas de conhecimento, no Gráfico 8 observa-se que a área de Gerenciamento de Projetos de Software é considerada a área com menor média de relevância (a média de relevância de seus tópicos

é de 10%). Porém, destaca-se que apenas 1 tópico desta área está nesse grupo. Por outro lado, todos os tópicos das áreas de Confiabilidade de Software e Métodos Formais são adotados por menos de 40% dos professores de Engenharia de Software.

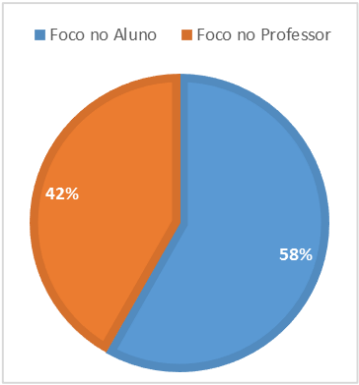
Gráfico 8 - Média de Tópicos Relevantes por Unidade de conhecimento



C) Abordagens de Ensino adotadas na disciplina de Engenharia de Software

Em relação aos métodos de ensino adotados pelos professores, há um certo equilíbrio, pois 42% dos professores adotam abordagens de ensino que focam nos alunos e 58% focam no professor, conforme Gráfico 9. Este enfoque está relacionado com Clima de aprendizagem, Programa de estudos, Objetivo de ensino, dentre outros.

Gráfico 9 - Métodos de Ensino adotados pelos Professores



Quanto as abordagens de ensino adotadas pelos professores, observa-se que todos adotam aulas expositivas e a maioria discute casos práticos com os alunos, realizam projetos de software e ministram aulas de laboratório. O percentual destas e de outras abordagens é apresentado no Gráfico 10.

Por fim, quanto as estratégias de avaliação adotadas pelos professores, observa-se que a grande maioria realiza trabalhos práticos, provas individuais e avaliam produtos de trabalhos gerados em projetos de software, conforme sintetiza o Gráfico 11.

Gráfico 10 - Abordagens de Ensino adotadas pelos Professores

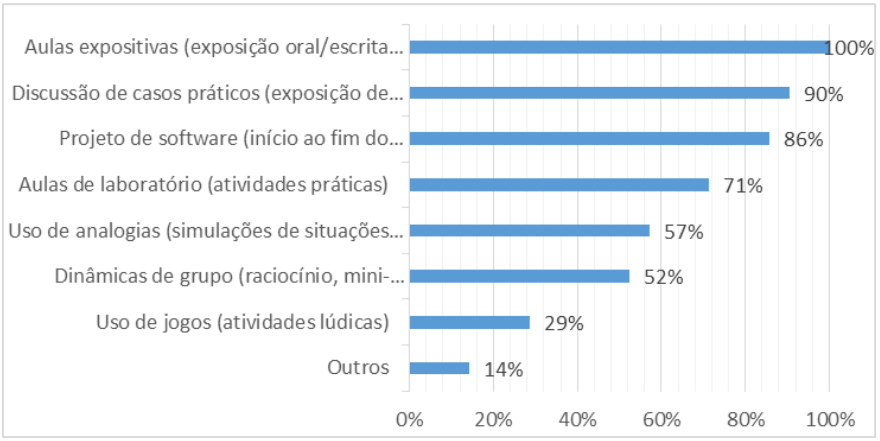
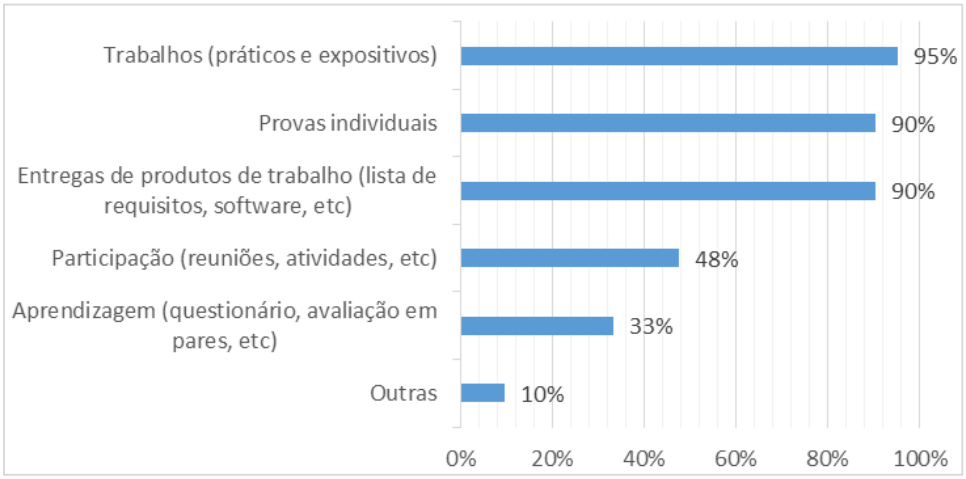


Gráfico 11 - Estratégias de Avaliação adotadas pelos Professores

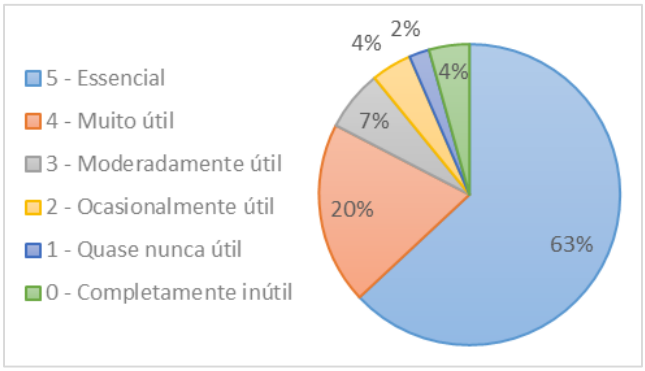


3.2. Alunos e a Aprendizagem dos Tópicos de Engenharia de Software

A) Utilidade da disciplina de Engenharia de Software

Em relação a percepção dos alunos quanto a utilidade da disciplina de Engenharia de Software para sua formação profissional, 63% considera a disciplina essencial, 20% muito útil e 7% moderadamente útil, conforme Gráfico 12. Observa-se que 10% dos alunos entrevistados não considera a disciplina muito útil.

Gráfico 12 - Percepção da Utilidade da Disciplina de Engenharia de Software

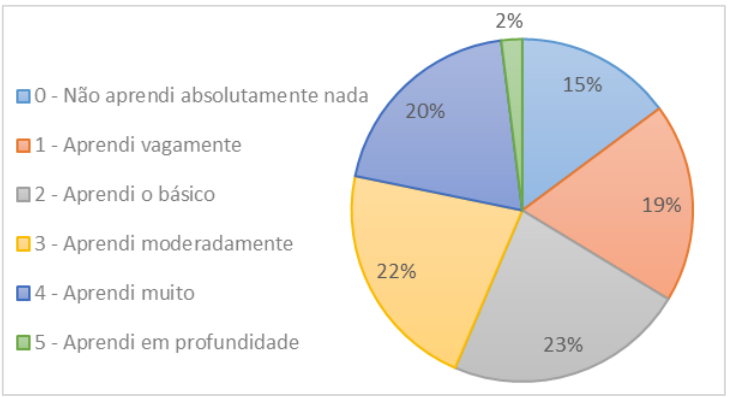


B) Aprendizagem dos Tópicos de Engenharia de Software

Assim como realizado com os dados dos questionários dos professores, apresentamos dois formatos complementares: listas categorizadas de tópicos e gráficos, retratando as 4 unidades de conhecimento com maior aprendizagem, de acordo com a percepção dos alunos, e as 4 unidades com menor aprendizagem. Para se chegar a estes números, consideram-se as aprendizagens que obtiveram maior porcentagem de grau 3, 4 e 5 na escala *Likert* (respectivamente "Aprendi muito" e "Aprendi em profundidade") as de maior aprendizagem, enquanto as aprendizagens que obtiveram maior porcentagem de grau 2, 1 e 0 na escala *Likert* são consideradas de menor aprendizagem. Além disto, para identificar as 4 de maior aprendizagem, consideram-se as unidades de conhecimento com média percentual de maior aprendizagem acima de 40% e as 4 de menor aprendizagem aquelas com média percentual de maior aprendizagem abaixo de 30%.

O Gráfico 13 apresenta o percentual de respostas obtidas para cada grau de aprendizagem. Observa-se um equilíbrio entre os graus Aprendi vagamente, Aprendi muito, Aprendi moderadamente e Aprendi o básico. Destaca-se também a pequena porcentagem de Aprendi em profundidade (2%) e a grande porcentagem de Não aprendi absolutamente nada (15%).

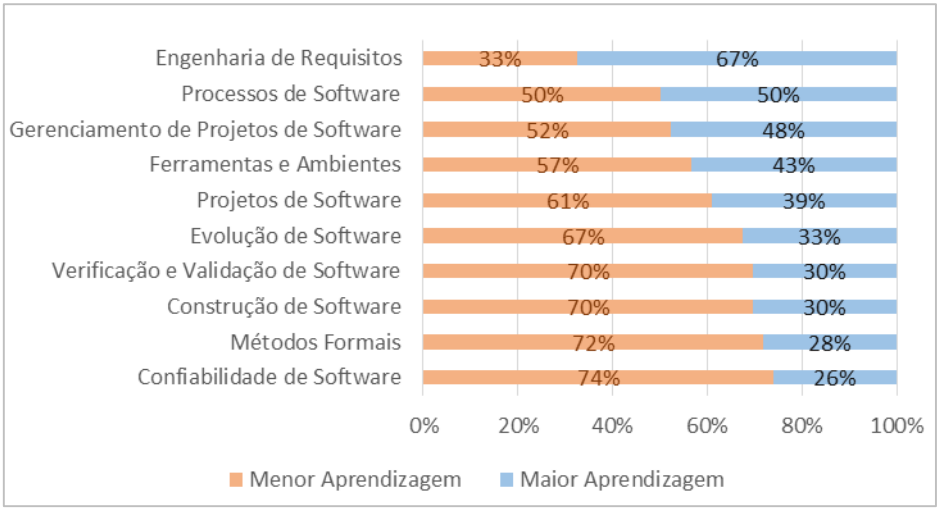
Gráfico 13 - Percentual de Graus de Aprendizagens dos Alunos



A Tabela 2 mostra informações sobre a aprendizagem esperada para cada unidade de conhecimento. Nesta tabela, o círculo azul identifica que a aprendizagem esperada é a que possui maior média de grau de aprendizagem para uma determinada unidade de conhecimento, enquanto o círculo laranja significa que a aprendizagem possui a menor média para uma unidade, de acordo com os alunos participantes do *survey*. Da mesma forma, o triângulo azul significa que a aprendizagem é uma das 18 mais aprendidas e o triângulo invertido com a cor laranja significa que é uma das 6 menos aprendidas.

Também é possível analisar a média percentual de aprendizagem para cada unidade de conhecimento. No Gráfico 14, observa-se que a Engenharia de Requisitos possui a maior porcentagem de aprendizagem, ou seja, 67% de seus tópicos são mais aprendidos pelos alunos.

Gráfico 14 - Média Percentual de Aprendizagem por Unidade de Conhecimento



Conforme a Tabela 4 e Gráfico 14, observa-se que, além da Engenharia de Requisitos, as unidades Processos de Software, Gerenciamento de Projetos de Software e Ferramentas e Ambientes possuem grande aprendizagem, acima de 40%. Já as unidades Verificação e Validação de Software, Construção de Software, Métodos Formais e Confiabilidade de Software apresentam menor aprendizagem, abaixo de 30%.

A partir da análise das aprendizagens de cada unidade, é possível observar a relevância de cada unidade de acordo com o percentual dos graus de suas aprendizagens esperadas. O Gráfico 15 apresenta as Unidades de Conhecimento da área de Engenharia de Software com maiores médias de grau de aprendizagem, segundo os alunos entrevistados. De todas as aprendizagens esperadas para a unidade de Engenharia de Requisitos, 92% possuem um alto grau de aprendizagem. Além de Engenharia de Requisitos, as unidades de Processos de Software, Gerenciamento de Projetos de Software e Ferramentas e Ambientes possuem grande aprendizagem, acima de 60%.

Gráfico 15 - Unidades de Conhecimento com Maiores Média de Aprendizagem

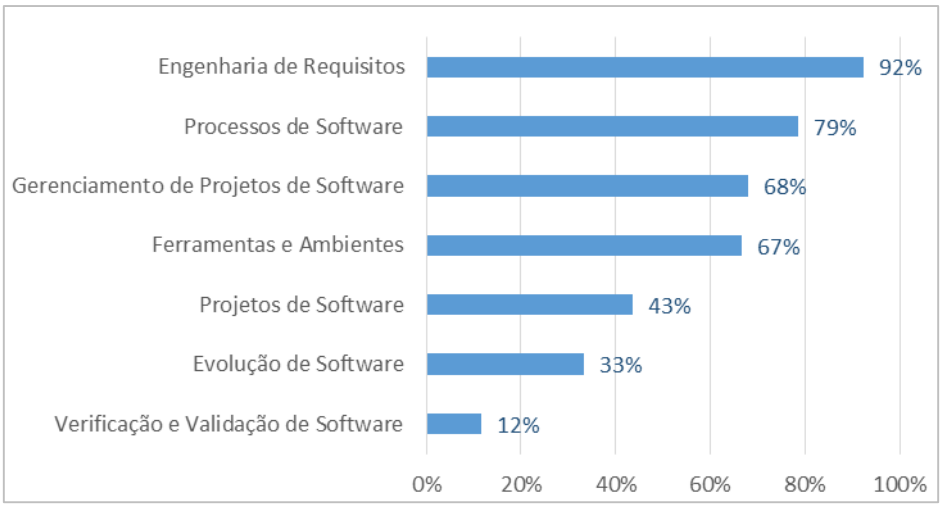


Tabela 4 - Aprendizagem Esperada para as Unidades de Conhecimento (ACM/IEEE e SBC)

Unidade de Conhecimento	Aprendizagem Esperada	% de Aprendizagem	Ranking Unidade	Ranking Área
Processos de Software	*1. Descrever como o software pode interagir e participar em vários sistemas.*	*70%*		▲
	2. Descrever as vantagens e desvantagens relativas entre os vários modelos de processos.	*67%*		▲
	3. Descrever as diferentes práticas que são componentes-chave de vários modelos de processos.	*46%*		▲
	4. Diferenciar as fases de desenvolvimento de software.	*83%*	●	▲
	5. Descrever como a programação em larga escala difere de esforços individuais no que diz respeito à compreensão de uma grande base de código, leitura de código, a compreensão sobre desenvolvimento, e compreensão do contexto de mudanças.	*50%*		▲
	6. Explicar o conceito de ciclo de vida de um software e fornecer um exemplo, ilustrando as suas fases, incluindo as entregas que são produzidos.	*80%*		▲
	7. Comparar vários modelos de processo no que diz respeito ao seu valor para o desenvolvimento de determinadas classes de sistemas de software, tendo em conta questões como a estabilidade exigência, tamanho e características não-funcionais.	*54%*		▲
	8. Definir a qualidade do software e descrever o papel das atividades de garantia da qualidade no processo de software.	*61%*		▲
	9. Descrever a intenção e semelhanças fundamentais entre as abordagens de melhoria de processos.	*50%*		▲
	10. Comparar vários modelos de melhoria de processos, tais como CMM, CMMI, CQI, Plan-Do-Check-Act ou ISO 9000.	*37%*	●	

Unidade de Conhecimento	Aprendizagem Esperada	% de Aprendizagem	Ranking Unidade	Ranking Área
	11. Avaliar um esforço de desenvolvimento e recomendar possíveis alterações ao participar de melhoria de processos (usando um modelo como PSP) ou a prática de uma retrospectiva do projeto.	*37%*	●	
	12. Explicar o papel de modelos de maturidade em processos de melhoria de processos.	*39%*		
	13. Descrever várias métricas de processo para avaliar e controlar um projeto.	*41%*		▲
	14. Usar métricas do projeto para descrever o estado atual de um projeto.	*43%*		▲
Gerenciamento de Projetos de Software	*1. Discutir comportamentos comuns que contribuem para o bom funcionamento de uma equipe.*	*63%*		▲
	2. Criar e seguir uma agenda para reunião de equipe.	*72%*	●	▲
	3. Identificar e justificar papéis necessários em uma equipe de desenvolvimento de software.	*72%*	●	▲
	4. Compreender as fontes, perigos e benefícios potenciais de conflito na equipe.	*70%*		▲
	5. Aplicar uma estratégia de resolução de conflitos em um ambiente de equipe.	*39%*		
	6. Utilizar um método ad hoc para estimar esforço de desenvolvimento de software (por exemplo, tempo) e comparar com o esforço real necessário.	*37%*		
	7. Listar vários exemplos de riscos de software.	*61%*		▲
	8. Descrever o impacto do risco em um ciclo de desenvolvimento de software.	*57%*		▲
	9. Descrever as diferentes categorias de risco em sistemas de software.	*43%*		▲
	10. Demonstrar através do envolvimento em uma equipe de projeto os elementos centrais da formação de equipes e gestão de equipe.	*39%*		

Unidade de Conhecimento	Aprendizagem Esperada	% de Aprendizagem	Ranking Unidade	Ranking Área
	11. Descrever como a escolha do modelo de processo afeta as estruturas organizacionais da equipe e processos de tomada de decisão.	*48%*		▲
	12. Criar uma equipe identificando papéis adequados e atribuir funções aos membros da equipe.	*70%*		▲
	13. Avaliar e fornecer feedback para equipes e indivíduos do seu desempenho em um ambiente de equipe.	*61%*		▲
	14. Usar um processo de software particular, descrever os aspectos de um projeto que precisam ser planejados e monitorados (por exemplo, as estimativas de tamanho e esforço, um cronograma, alocação de recursos, controle de configuração, gerenciamento de mudanças, e identificação de riscos de projetos e gestão).	*63%*		▲
	15. Acompanhar o progresso de algum estágio de um projeto usando métricas de projeto adequadas.	*48%*		▲
	16. Comparar técnicas de tamanho e estimativa de software de custos simples.	*48%*		▲
	17. Usar uma ferramenta de gerenciamento de projetos para auxiliar na atribuição e acompanhamento de tarefas em um projeto de desenvolvimento de software.	*52%*		▲
	18. Descrever o impacto de tolerância ao risco no processo de desenvolvimento de software.	*37%*		
	19. Identificar os riscos e descrever abordagens para a gestão dos riscos (prevenção, a aceitação, a transferência, mitigação), e caracterizar os pontos fortes e fracos de cada um.	*50%*		▲
	20. Explicar como o risco afeta as decisões no processo de desenvolvimento de software.	*48%*		▲
	21. Identificar os riscos de segurança de um sistema de software.	*43%*		▲

Unidade de Conhecimento	Aprendizagem Esperada	% de Aprendizagem	Ranking Unidade	Ranking Área
	22. Demonstrar uma abordagem sistemática para a tarefa de identificar os perigos e riscos em uma situação particular.	*33%*		
	23. Aplicar os princípios básicos de gestão de risco em uma variedade de cenários simples, incluindo a situação de segurança.	*30%*		▼
	24. Realizar uma análise de custo/benefício para uma abordagem de mitigação de risco.	*28%*	●	▼
	25. Identificar e analisar alguns dos riscos para todo um sistema que surgem a partir de outros aspectos.	*37%*		
Ferramentas e Ambientes	*1. Descrever a diferença entre a gestão de configuração de software centralizada e distribuída.*	*37%*	●	
	2. Descrever como controle de versão pode ser usado para ajudar a controlar o gerenciamento de versão de software.	*54%*	●	▲
	3. Identificar os itens de configuração e usar uma ferramenta de controle de código-fonte em um pequeno projeto baseado em equipe.	*46%*		▲
	4. Descrever como as ferramentas de teste estático e dinâmico disponíveis podem ser integradas ao ambiente de desenvolvimento de software.	*37%*	●	
	5. Descrever as questões que são importantes na escolha de um conjunto de ferramentas para o desenvolvimento de um sistema de software particular, incluindo ferramentas para rastreamento de requisitos, modelagem de projeto, implementação, construção automática e testes.	*41%*		▲
	6. Demonstrar a capacidade de usar ferramentas de software de apoio ao desenvolvimento de um produto de software de tamanho médio.	*46%*		▲
Engenharia de Requisitos	*1. Listar os principais componentes de um caso de uso ou descrição semelhante de algum comportamento que é necessário para um sistema.*	*85%*	●	▲
	2. Descrever como o processo de engenharia de requisitos apoia a elicitação e validação de requisitos comportamentais.	*85%*	●	▲

Unidade de Conhecimento	Aprendizagem Esperada	% de Aprendizagem	Ranking Unidade	Ranking Área
	3. Interpretar um determinado modelo de requisitos para um sistema de software simples.	*76%*		▲
	4. Descrever os desafios fundamentais de técnicas comuns usadas para elicitação de requisitos.	*72%*		▲
	5. Listar os principais componentes de um modelo de dados (por exemplo, diagramas de classe ou diagramas ER).	*85%*	●	▲
	6. Identificar os requisitos funcionais e não-funcionais numa determinada especificação de requisitos para um sistema de software.	*80%*		▲
	7. Realizar uma avaliação de um conjunto de requisitos de software para determinar a qualidade dos requisitos no que diz respeito às boas características de requisitos.	*67%*		▲
	8. Aplicar elementos-chave e métodos comuns para elicitação e análise para produzir um conjunto de requisitos para um sistema de software de médio porte.	*61%*		▲
	9. Comparar as abordagens baseadas em plano e abordagens ágeis para a especificação e validação de requisitos e descrever os benefícios e os riscos associados a cada um.	*54%*		▲
	10. Usar um método comum, não-formal para modelar e especificar os requisitos para um sistema de software de médio porte.	*54%*		▲
	11. Traduzir em linguagem natural uma especificação de requisitos de software (por exemplo, um contrato de componente de software) escrito em uma linguagem de especificação formal.	*46%*		▲
	12. Criar um protótipo de um sistema de software para limitar os riscos nos requisitos.	*61%*		▲
	13. Diferenciar entre rastreabilidade a frente e para trás e explicar as suas funções no processo de validação de requisitos.	*39%*	●	

Unidade de Conhecimento	Aprendizagem Esperada	% de Aprendizagem	Ranking Unidade	Ranking Área
Projetos de Software	*1. Articular os princípios de projeto, incluindo a separação de interesses, a ocultação de informações, o acoplamento e coesão, e encapsulamento.*	*48%*		▲
	2. Usar um padrão de projeto para projetar um sistema de software simples, e explicar como princípios de padrões foram aplicadas neste projeto.	*48%*		▲
	3. Construir modelos do projeto de um sistema de software simples que são apropriadas para o paradigma utilizado para projetá-lo.	*43%*		▲
	4. Dentro do contexto de um único paradigma de projeto, descrever um ou mais padrões de projeto que também possam ser aplicados para a concepção de um sistema de software simples.	*43%*		▲
	5. Para um sistema simples adequado para um determinado cenário, discutir e selecionar um padrão de projeto apropriado.	*46%*		▲
	6. Criar modelos adequados para a estrutura e o comportamento de produtos de software a partir das suas especificações de requisitos.	*59%*	●	▲
	7. Explicar as relações entre os requisitos para um produto de software e seu padrão de projeto, utilizando modelos apropriados.	*46%*		▲
	8. Para o padrão de projeto de simples sistema de software dentro do contexto de um único paradigma de projeto, descrever a arquitetura de software de sistema.	*50%*		▲
	9. Dado um projeto de alto nível, identificar a arquitetura de software através da diferenciação entre arquiteturas de software comuns, tais como 3-tier, pipe-and-filter, e cliente-servidor.	*37%*		
	10. Investigar o impacto da seleção de arquiteturas de software sobre a concepção de um sistema simples.	*39%*		
	11. Aplicar exemplos simples de padrões em um projeto de software.	*50%*		▲
	12. Descrever a forma de refatoração e discutir em que pode ser aplicável.	*39%*		

Unidade de Conhecimento	Aprendizagem Esperada	% de Aprendizagem	Ranking Unidade	Ranking Área
	13. Selecionar componentes adequados para utilização na concepção de um produto de software.	*37%*		
	14. Explicar como componentes adequados podem precisar ser adaptados para o uso no padrão de um produto de software.	*28%*		▼
	15. Projetar um contrato para um componente de software pequeno típico para uso em um determinado sistema.	*37%*		
	16. Discutir e selecionar a arquitetura de software apropriado para um sistema simples adequado para um determinado cenário.	*43%*		▲
	17. Aplicar modelos para qualidades internas e externas na concepção de componentes de software para conseguir uma troca aceitável entre aspectos conflitantes de qualidade.	*39%*		
	18. Analisar um projeto de software a partir da perspectiva de um atributo de qualidade interna significativa.	*35%*		
	19. Analisar um projeto de software a partir da perspectiva de um atributo de qualidade externo significativo.	*33%*		
	20. Explicar o papel de objetos em sistemas de middleware e a relação com os componentes.	*28%*		▼
	21. Adotar medidas orientadas a componentes para a concepção de uma gama de software, tais como o uso de componentes para a simultaneidade e operações, para os serviços de comunicação confiáveis, para a interação de banco de dados ou para comunicação segura e de acesso.	*28%*		▼
	22. Refatorar uma implementação de software existente para melhorar algum aspecto de seu projeto.	*33%*		
	23. Aplicar os princípios de privilégio mínimo e padrões de falhas-seguras.	*26%*	●	▼

Unidade de Conhecimento	Aprendizagem Esperada	% de Aprendizagem	Ranking Unidade	Ranking Área
Construção de Software	*1. Descrever as técnicas de codificação, expressões idiomáticas e mecanismos para a implementação de projetos para alcançar as propriedades desejadas, tais como confiabilidade, eficiência e robustez.*	*37%*		
	2. Construir código robusto usando mecanismos de manipulação de exceção.	*30%*		▼
	3. Descrever codificação segura e práticas de codificação defensivas.	*24%*		▼
	4. Selecionar e usar um padrão de codificação definido em um projeto de software pequeno.	*39%*	●	
	5. Comparar e contrastar estratégias de integração, incluindo top-down, bottom-up, e integração sanduíche.	*22%*	●	▼
	6. Descrever o processo de análise e implementação de alterações à base de código desenvolvido para um projeto específico.	*33%*		
	7. Descrever o processo de análise e implementação de mudanças para uma grande base de código existente.	*24%*		▼
	8. Reescrever um programa simples para remover as vulnerabilidades comuns, como estouros de buffer, overflows, inteiros e condições de corrida.	*30%*		▼
	9. Fazer um componente de software que execute alguma tarefa não-trivial e que seja resistente a erros de entrada e tempo de execução.	*24%*		▼
Verificação e Validação de Software	*1. Distinguir entre a validação e verificação do programa.*	*52%*	●	▲
	2. Descrever o papel que as ferramentas podem ter na validação de software.	*39%*		
	3. Realizar, como parte de uma atividade de equipe, uma inspeção a um segmento de código de tamanho médio.	*33%*		
	4. Descrever e distinguir entre os diferentes tipos e níveis de testes (unidade, integração, sistemas e aceitação).	*41%*		▲

Unidade de Conhecimento	Aprendizagem Esperada	% de Aprendizagem	Ranking Unidade	Ranking Área
	5. Descrever as técnicas para a identificação de casos de teste significativos para a integração, regressão e teste do sistema.	*30%*		▼
	6. Criar e documentar um conjunto de testes para um segmento de código de tamanho médio.	*33%*		
	7. Descrever como selecionar bons testes de regressão e automatizá-las.	*20%*	●	▼
	8. Usar uma ferramenta de rastreamento de defeitos para gerir os defeitos de software em um projeto de software pequeno.	*24%*		▼
	9. Discutir as limitações do teste em um domínio particular.	*26%*		▼
	10. Avaliar uma suíte de testes para um segmento de código de tamanho médio.	*30%*		▼
	11. Comparar as abordagens estáticas e dinâmicas para verificação.	*33%*		
	12. Identificar os princípios fundamentais de métodos de desenvolvimento orientado a testes e explicar o papel do teste automatizado nestes métodos.	*35%*		
	13. Discutir as questões que envolvem o teste de software orientado a objetos.	*28%*		▼
	14. Descrever as técnicas para a verificação e validação de artefatos não-código.	*26%*		▼
	15. Descrever as abordagens para a estimativa de falhas.	*35%*		
	16. Estimar o número de falhas em um aplicativo de software pequeno com base na densidade de falhas.	*20%*	●	▼
	17. Proceder a uma inspeção ou revisão de código-fonte do software para um projeto de software pequeno ou média.	*28%*		▼
Evolução de Software	*1. Identificar os principais problemas associados à evolução do software e explicar seu impacto sobre o ciclo de vida do software.*	*46%*	●	▲

Unidade de Conhecimento	Aprendizagem Esperada	% de Aprendizagem	Ranking Unidade	Ranking Área
	2. Estimar o impacto de uma solicitação de mudança para um produto já existente, de tamanho médio.	*33%*		
	3. Usar refatoração no processo de modificação de um componente de software.	*26%*		▼
	4. Discutir os desafios dos sistemas de evolução em um ambiente em mudança.	*33%*		
	5. Resumir o processo de testes de regressão e seu papel na gestão do release.	*15%*	●	▼
	6. Discutir as vantagens e desvantagens dos diferentes tipos de reuso de software.	*43%*		▲
Confiabilidade de Software	*1. Explicar os problemas que se colocam ao atingimento de altos níveis de confiabilidade.*	*20%*	●	▼
	2. Descrever como a confiabilidade do software contribui para a confiabilidade do sistema.	*30%*		▼
	3. Listar abordagens para minimizar falhas que podem ser aplicadas em todas as fases do ciclo de vida do software.	*37%*	●	
	4. Comparar as características de três diferentes abordagens de modelagem de confiabilidade.	*26%*		▼
	5. Demonstrar a capacidade de aplicar vários métodos para desenvolver estimativas de confiabilidade para um sistema de software.	*28%*		▼
	6. Identificar métodos que levem à realização de uma arquitetura de software que permite atingir um determinado nível de confiabilidade.	*24%*		▼
	7. Identificar formas de aplicar redundância para alcançar tolerância a falhas para uma aplicação de médio porte.	*24%*		▼

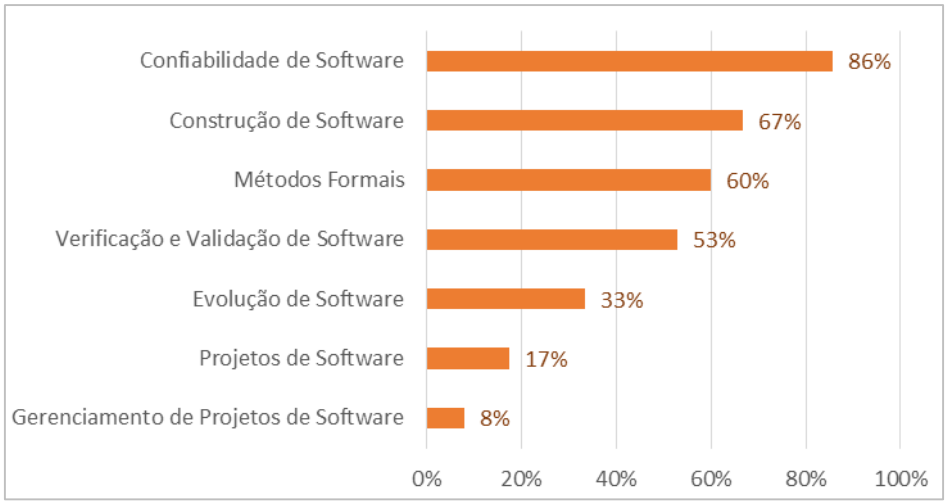
Unidade de Conhecimento	Aprendizagem Esperada	% de Aprendizagem	Ranking Unidade	Ranking Área
Métodos Formais	*1. Descrever o papel que técnicas de análise e especificação pode desempenhar no desenvolvimento de software complexo e comparar a sua utilização como técnicas de validação e verificação com o teste.*	*26%*		▼
	2. Aplicar técnicas formais de especificação e análise de projetos de software e programas de baixa complexidade.	*33%*		
	3. Explicar os potenciais benefícios e desvantagens do uso de linguagens formais de especificação.	*28%*		▼
	4. Criar e avaliar as afirmações do programa para uma variedade de comportamentos que vão desde simples até complexos.	*20%*	●	▼
	5. Usando uma linguagem de especificação formal comum, formular a especificação de um sistema de software simples e derivar exemplos de casos de teste a partir da especificação.	*37%*	●	

Legenda: Aprendizagens com Maior Média da Unidade de Conhecimento (●) Aprendizagens com Menor Média da Unidade de Conhecimento (●)

Aprendizagens com Maior Média da Engenharia de Software (▲) Aprendizagens com Menor Média da Engenharia de Software (▼)

Já o Gráfico 16 apresenta as Unidades de Conhecimento da área de Engenharia de Software com as menores médias de grau de aprendizagem, segundo os alunos entrevistados. De todas as aprendizagens esperadas para a unidade de Confiabilidade de Software, 86% possuem um baixo grau de aprendizagem. Além de Confiabilidade de Software, as unidades de Construção de Software, Métodos Formais e Verificação e Validação de Software possuem um baixo grau de aprendizagem, acima de 50%.

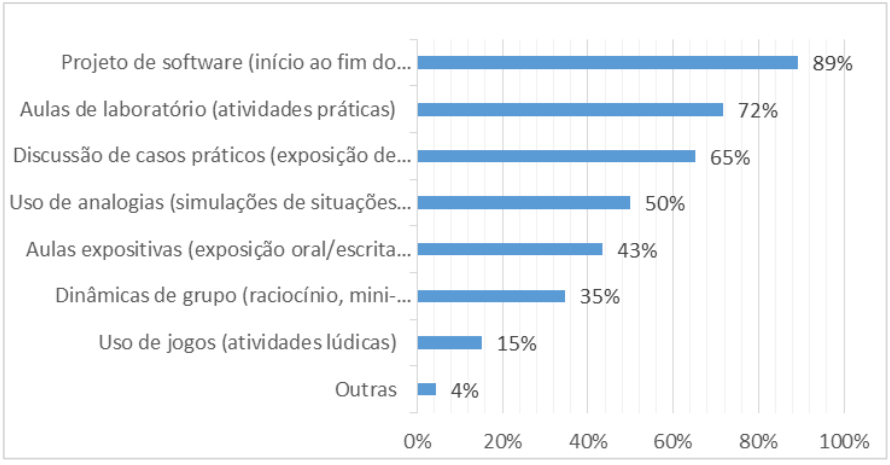
Gráfico 16 - Unidades de Conhecimento com Menores Média de Aprendizagem



C) Abordagens de Ensino Consideradas Efetivas

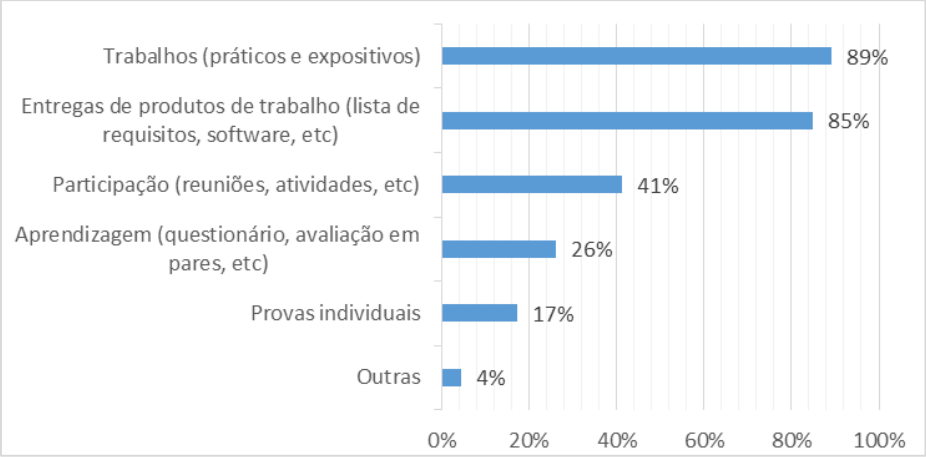
Quanto as abordagens de ensino consideradas efetivas pelos alunos, observa-se que a grande maioria considera a realização de Projetos de software, Aulas de laboratório e Discussão de casos práticos. O percentual destas e de outras abordagens é apresentado no Gráfico 17.

Gráfico 17 - Percepção de Efetividade das Abordagens de Ensino



Por fim, quanto as estratégias de avaliação adotadas pelos professores, observa-se que a grande maioria dos alunos considera efetivo a realização de trabalhos práticos e entrega de produtos de trabalhos gerados em projetos de software, conforme sintetiza o Gráfico 18.

Gráfico 18 - Percepção de Efetividade das Estratégias de Avaliação



4. DISCUSSÕES SOBRE O *SURVEY*

4.1. Sobre a Adoção de Currículos de Referência

Em relação à adoção de currículos de referência, 24% dos professores entrevistados adota o currículo de referência da SBC, 16% utiliza o currículo da ACM/IEEE e outros 16% adota estes 2 currículos. Sendo assim, observa-se que 56% dos professores adota um currículo de referência para definir suas ementas da disciplina de Engenharia de Software. É um baixo percentual, considerando que 16% adota ementas definidas pela própria instituição, 12% considera outras abordagens e 16% dos professores não adota nenhum currículo de referência na definição de suas ementas. Isto totaliza 44% de professores que não adota currículos de referência.

É de suma importância a adoção destes currículos, pois são discutidos e elaborados por órgãos representativos da área como a Sociedade Brasileira da Computação (SBC) e ACM/IEEE. Além disso, definem uma estrutura de conceitos inter-relacionados a fim de especificar diretrizes de ensino e formação profissional de acordo com as áreas da Computação, sendo uma delas a Engenharia de Software. Estas diretrizes definem o perfil profissional e acadêmico esperado para os estudantes da área, bem como estruturação e detalhamento de matérias, como carga horária, tópicos a serem abordados e aprendizagens esperadas para cada um destes tópicos.

Sem a adoção destes currículos, os professores acabam por estabelecer ementas incompatíveis com diretrizes nacionais e internacionais de ensino e, possivelmente, não atendendo o que se espera de um curso de graduação em computação. Como não há diretrizes curriculares, não é possível comparar a qualidade da ementa destes professores com o restante daqueles que adotam estes currículos da SBC e ACM/IEEE.

4.2. Sobre a Utilidade e a Aprendizagem da Disciplina de Engenharia de Software

Em relação à percepção da utilidade da disciplina de Engenharia de Software, 90% dos alunos entrevistados consideram esta útil, ou seja, responderam Essencial, Muito Útil ou Moderadamente útil. De acordo com a ACM/IEEE, a Engenharia de Software se

constitui como uma das disciplinas de maior relevância nos cursos da área de Computação. Isto decorre tanto da importância do processo de software, objeto de estudo desta disciplina, em si quanto dos desafios relacionados com a formação completa de um profissional que irá atuar no mercado de software. Grande parcela destes alunos, após a formação, irá atuar em um contexto de processo de desenvolvimento de software, sendo necessário conhecimento relacionado aos conceitos e tópicos relacionados a esta área.

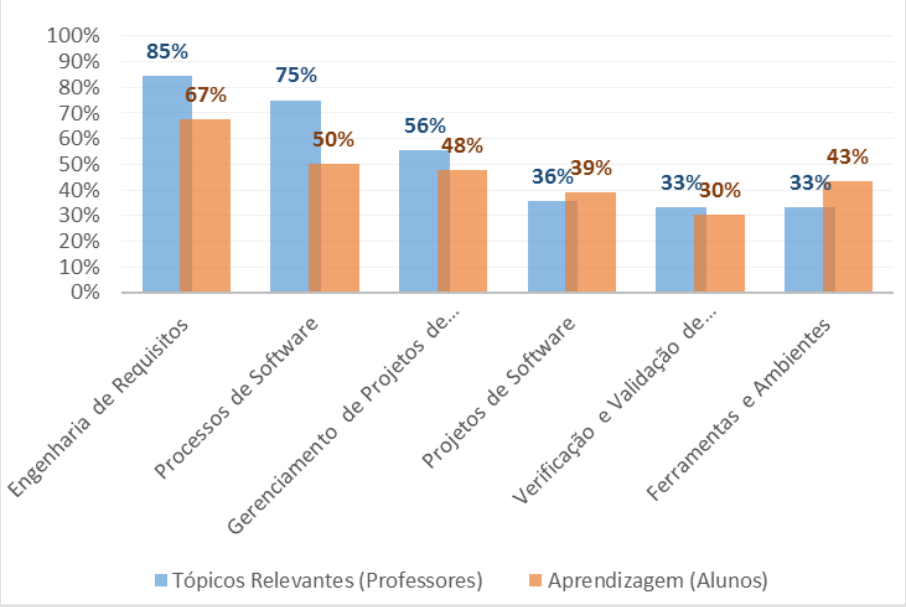
No que diz respeito à aprendizagem, os alunos apresentaram um baixo percentual para a resposta "Aprendi em profundidade" (apenas 2%) e uma grande porcentagem para "Não aprendi absolutamente nada" (15% das respostas dos alunos). Isto acaba por reforçar as constatações da indústria de que os alunos não estão tendo uma formação adequada. Em particular, na Engenharia de Software, a grande quantidade de tópicos acaba por favorecer este cenário, onde os alunos não conseguem profundidade no aprendizado de determinados tópicos.

4.3. Sobre a Relevância dos Tópicos

A) Os Mais Relevantes

De 10 unidades de conhecimento da Engenharia de Software, destacam-se as 6 que possuem tópicos mais relevantes, ou seja, aqueles que são mais adotados pelos professores nas ementas das disciplinas. A partir da identificação destas unidades, é possível correlacionar o percentual de tópicos relevantes com o percentual de aprendizagem dos alunos, conforme Gráfico 19.

Gráfico 19 - Correlação entre Tópicos Mais Relevantes e Aprendizagem



Observa-se que a unidade Engenharia de Requisitos, de acordo com os resultados deste *survey*, é que possui maior relevância, pois é amplamente contemplada nos currículos de Engenharia de Software, por 85% dos professores, e efetivamente aprendida por 67% dos alunos entrevistados. Em seguida, destacam-se as unidades de Processos de Software, ministrada por 75% dos professores e aprendida por 50% dos alunos, e

Gerenciamento de Projetos de Software, ministrada por 56% dos professores e aprendida por 48% dos alunos entrevistados. Para as demais unidades relevantes, ouve um certo desalinhamento entre relevância e aprendizagem. Por exemplo, 36% dos professores abordam Projetos de Software, porém 39% dos alunos aprendem efetivamente esta unidade, um percentual menor do que a aprendizagem da unidade Ferramentas e Ambientes, 43%, que é abordada apenas por 33% dos professores.

Acredita-se que a relevância da unidade Engenharia de Requisitos deve-se ao fato de que a Engenharia de Software é uma disciplina preocupada com o desenvolvimento efetivo e eficiente de sistemas de software que satisfaçam os requisitos dos usuários. Para satisfazer as necessidades dos usuários, é de extrema importância o conhecimento de técnicas de elicitação, modelagem e escrita de requisitos, tópicos estes abordados nesta unidade.

Além disto, os profissionais 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. Neste contexto, insere-se a segunda unidade mais relevante, Processos de Software, o principal objeto de estudo da Engenharia de Software. De acordo com as aprendizagens esperadas para esta unidade, os alunos devem ser capazes de definir, modelar e executar um processo de software.

No entanto, não basta definir um processo e um plano de projeto, é necessário pôr em prática em fazer o acompanhamento deste para realizar ajustes caso necessário. Neste contexto, se insere a terceira unidade mais relevante, Gerenciamento de Projetos de Software. O Projeto de Software consiste em instanciar um Processo a fim de atender um plano específico, que detalha um escopo, cronograma, equipe e riscos. A relevância desta unidade deve-se a importância e dificuldade de realizar a gerência de um Projeto e, principalmente, da equipe que irá executá-lo. Observa-se, assim, que a gerência se mostra mais importante que o próprio projeto, segundo os professores e alunos entrevistados.

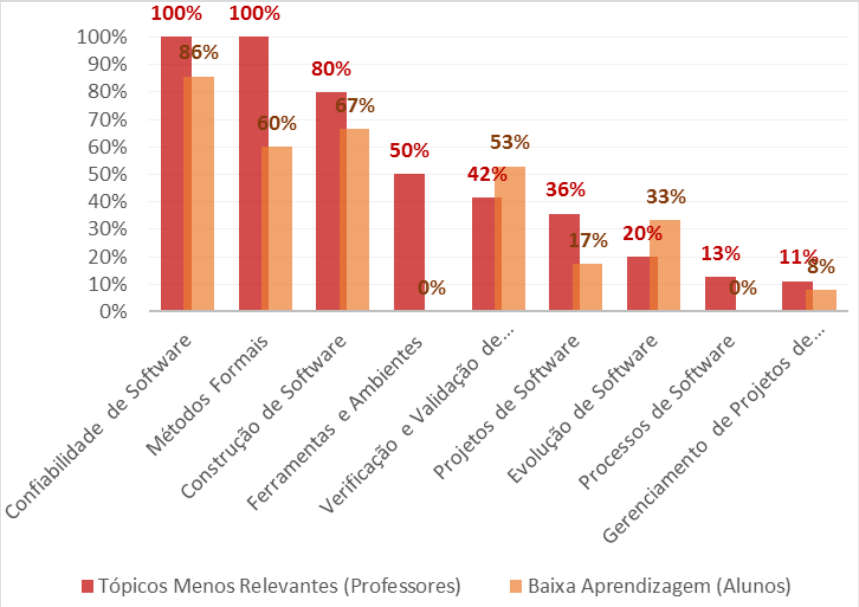
Por fim, quanto as três unidades restantes, observa-se uma complementariedade. A unidade de Projetos de Software consiste em adotar padrões de projetos, arquiteturas, interfaces, qualidade a fim de atender um conjunto específico de Requisitos seguindo um Processo de Software. Como dito anteriormente, o Gerenciamento de Projetos é a unidade responsável pela sua execução e acompanhamento. A unidade de Verificação e Validação complementa a unidade de Projeto na medida em que é responsável pelos testes e auditorias de qualidade do processo executado e dos produtos de trabalho gerados. Por fim, a unidade Ferramentas e Ambientes apoia a definição, modelagem e execução de Processos, bem como a execução de Projetos na medida que permite realizar testes e controle de versões, por exemplo.

B) Os Menos Relevantes

De acordo com este *survey*, 4 unidades de conhecimento foram consideradas menos relevantes, ou seja, aquelas que são menos contempladas nas ementas de Engenharia de Software pelos professores entrevistados. São estas Confiabilidade de Software, Métodos Formais, Construção de Software e Evolução de Software. Porém, em

relação à aprendizagem dos alunos, 9 unidades apresentam baixa aprendizagem. É possível correlacionar o percentual de tópicos menos relevantes com o percentual de baixa aprendizagem dos alunos, conforme Gráfico 20.

Gráfico 20 - Correlação entre Tópicos Menos Relevantes e Aprendizagem



Destacam-se as 3 unidades Confiabilidade de Software, Métodos Formais e Construção de Software, as 2 primeiras com 100% de tópicos pouco abordados pelos professores e a terceira com 80% de baixa adoção. O fato destas unidades serem pouco relevantes para os professores impacta diretamente na aprendizagem dos alunos, que demonstraram baixa aprendizagem, 86%, 60% e 67% respectivamente, para os tópicos ministrados nestas unidades. A quarta unidade menos abordada pelos professores, Evolução de Software, possui 20% de tópicos considerados menos relevantes e 33% de baixa aprendizagem em relação a estes.

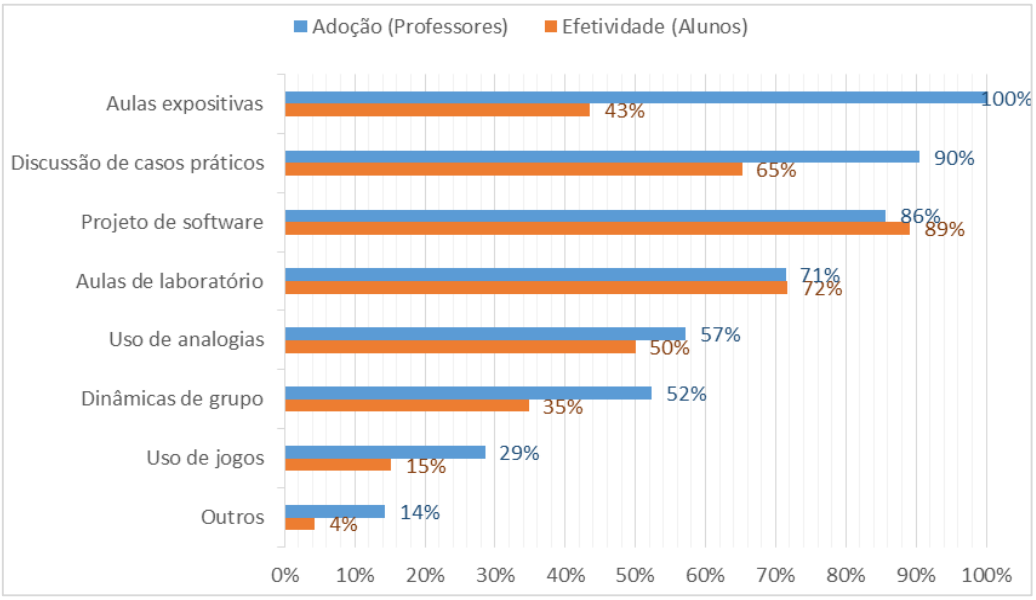
Apesar de consideradas relevantes, as unidades Ferramentas e Ambientes e Processos de Software também possuem tópicos considerados menos relevantes. No entanto, não apresentaram baixa aprendizagem pelos alunos. O mesmo não ocorre para as unidades Verificação e Validação de Software, Projetos de Software e Gerenciamento de Projetos de Software, que possuem índices equivalentes, ou superior, entre tópicos menos relevantes e baixa aprendizagem. Neste caso, é necessária uma atenção maior para o ensino dos tópicos destas 3 unidades.

4.4. Sobre as Abordagens de Ensino

Em relação aos métodos de ensino, destaca-se que aulas centradas no professor geralmente são mais expositivas. Uma aula expositiva acaba sendo pouco eficiente, pois ativa apenas o sentido da audição. Por outro lado, quanto mais prática e dinâmica for a aula, mais ela será centrada no aluno. A aprendizagem mais prática acaba sendo caracterizada pelo envolvimento dos alunos em determinadas atividades, com resultados mais positivos.

Quanto as abordagens de ensino, estabelece-se uma correlação entre a adoção destas pelos professores e o quanto os alunos consideram estas efetivas para seu aprendizado. O Gráfico 21 apresenta o resultado desta correlação.

Gráfico 21 - Correlação entre Abordagens de Ensino Adotadas e Percepção de Efetividade



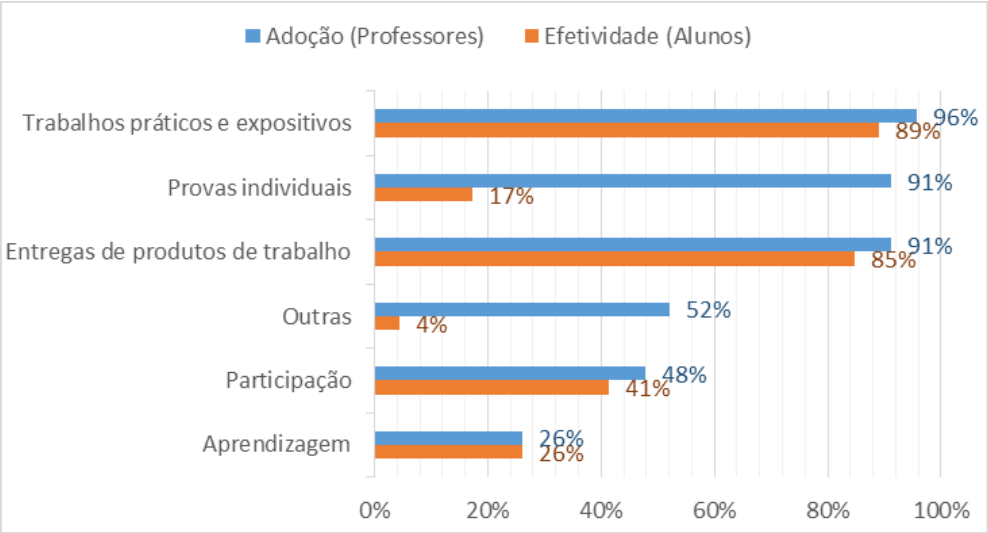
Apesar de todos professores realizarem Aulas expositivas, apenas 43% dos alunos consideram estas efetivas para seu aprendizado. É uma porcentagem menor do que a atribuída para Uso de analogias, adotada por menos de 70% dos professores. Também houve uma certa discordância em relação à Discussão de casos práticos, adotado por 90% dos professores e considerado efetivo por 65% dos alunos entrevistados. Houve convergência nas abordagens de Projeto de software e Aulas de laboratório.

A partir dos resultados deste *survey*, observa-se que os alunos estão interessados em realizar atividades práticas, como projetos de desenvolvimento em laboratórios que simulem situações próximas as que vão encontrar no mercado. Acredita-se que isso se deve ao fato da disciplina Engenharia de Software possuir muitos tópicos, 83 no total, o que acaba por torná-la menos atrativa para alunos que não possuem uma afinidade com a área. A abordagem prática permite aos alunos fixar melhor os conceitos a partir da sua aplicação.

Em relação as estratégias de avaliação, realiza-se a mesma correlação entre a adoção pelos professores e a percepção de efetividade para o aprendizado dos alunos, conforme apresenta o Gráfico 22.

Destaca-se que a maioria dos alunos não consideram efetivas as tradicionais provas individuais, adotadas por 91% dos professores entrevistados. Há também uma baixa aceitação de outras técnicas, apenas 4% dos alunos. Acredita-se que isso se deve ao fato dos alunos não saberem especificamente, no questionário da pesquisa, quais seriam estas estratégias, pois este era um campo aberto (descritivo).

Gráfico 22 - Correlação entre Estratégias de Avaliação Adotadas e Percepção de Efetividade



Houve convergência nas estratégias de Trabalhos práticos e expositivos, Entrega de produtos de trabalho, Participação e Aprendizagem. Há uma preferência pelas estratégias de avaliação práticas, reforçando o interesse dos alunos por abordagens de ensino práticas.

5. CONCLUSÕES

5.1. Resumo

Este relatório apresenta os resultados de um *survey* realizado com professores e alunos de Engenharia de Software de instituições públicas e privadas do Brasil. Foram entrevistados 23 professores e 47 alunos, distribuídos nas cinco regiões do Brasil. No total, o *survey* alcançou 12 estados, sendo a maioria destes do Nordeste, 5 estados e 10 cidades. Os entrevistados representam 20 instituições de ensino superior, sendo 80% pública e 20% privada.

Observa-se que o professor que possui o ano de formação mais antigo foi 1998 e o que leciona a disciplina de Engenharia de Software há mais tempo foi 25 anos. A média de idade destes professores é de 38 anos. Quanto aos alunos, a média de idade é 24 anos e o ano de conclusão da disciplina mais antigo foi 2005, atendendo ao critério de inclusão da pesquisa (ano de lançamento do currículo de referência da SBC).

A unidade de conhecimento considerada mais relevante foi a Engenharia de Requisitos, enquanto Métodos Formais e Confiabilidade de Software foram consideradas as menos relevantes para o ensino de Engenharia de Software.

Quanto aos métodos de ensino, observa-se uma preferência, por grande parte dos alunos entrevistados, por abordagens práticas de ensino e avaliação, como Projeto de software, Aulas de laboratório, Trabalhos práticos e expositivos e Entrega de produtos de trabalho.

5.2. Resultados Obtidos

Os resultados obtidos a partir da condução deste *survey* são de alta relevância para a área de Engenharia de Software. Portanto, os pesquisadores e profissionais que atuam nesta área necessitam de mais estudos empíricos e, consequentemente, esclarecimentos em relação ao ensino desta disciplina a fim de compreender como estar sendo formado o profissional da área.

O cenário atual do ensino demonstra que determinados tópicos são considerados de baixa relevância, pelos professores e, consequentemente, apresentam baixo grau de aprendizagem pelos alunos. Acontece que estes tópicos consomem carga horária considerável da disciplina de Engenharia de Software, enquanto que alguns tópicos considerados relevantes apresentam baixa aprendizagem. Talvez isso deva-se ao fato de não haver tempo hábil para ministrar de maneira efetiva todos os tópicos destas unidades relevantes.

Analisando os resultados obtidos, acredita-se que as 6 unidades de conhecimento consideradas relevantes são totalmente complementares, sendo possível correlacionar seus tópicos. Essa correlação se apresenta como um trabalho futuro desta pesquisa, onde definir-se-á uma ementa baseada nestas unidades de conhecimento. Para as estratégias de ensino, deve-se ter uma atenção maior para os tópicos de Verificação e Validação de Software, Projetos de Software e Gerenciamento de Projetos de Software que apesar de considerados relevantes, apresentaram um baixo índice de aprendizagem pelos alunos entrevistados.

Quanto as unidades consideradas menos relevantes, acredita-se que o resultado se deve ao fato destas serem focos de outras disciplinas. Por exemplo, Construção e Evolução de Software são abordadas em diversas disciplinas relacionadas ao desenvolvimento de software a partir de uma determinada linguagem e/ou paradigma, como Java, C++, Orientação a Objetos, Web, dentre outros. Já a unidade Confiabilidade de Software, indiretamente em disciplinas de redes, relacionada a tópicos de tolerância a falhas. Por fim, para a unidade de Métodos Formais, tradicionalmente há uma disciplina específica em cursos de Ciência da Computação.

Analisando os métodos de ensino, observa-se uma convergência entre alunos e professores no que diz respeito a adoção de abordagens práticas de ensino e avaliação. O caráter da disciplina de Engenharia de Software, tradicionalmente teórico, faz com que estes prefiram aplicar os tópicos sugeridos pelos currículos de referência através de Projetos de software, onde o aluno é avaliado pela entrega de produtos de trabalho gerados (como Lista de Requisitos, Diagramas UML, código-fonte, dentre outros) e/ou Aulas de laboratório, onde os alunos apresentam trabalhos práticos e expositivos. Esta constatação destaca a importância de se adotar abordagens de ensino imersivas, como a *Problem-Based Learning* (PBL), o uso de clientes do mercado, a definição de cronogramas e prazos e a atribuição de papéis e responsabilidades aos alunos. Além disso, considera-se importante adotar estratégias de avaliação reflexivas, como definição de mapas conceituais e avaliação em pares.

Conclui-se, a partir da análise da relevância das unidades e do fato de que os alunos estão demonstrando uma baixa porcentagem de "aprendizagem em profundidade" e uma alta porcentagem para "não aprendi nada", que o ensino de Engenharia de Software deva focar nas 6 unidades mais relevantes, considerando que estas são complementares e se enquadram perfeitamente no cenário de desenvolvimento de um projeto de software, abordagem considerada mais efetivas pelos alunos. Além disto, a redução da quantidade de tópicos permitirá um maior aprofundamento no desenvolvimento de competências e habilidades profissionais relacionadas aos tópicos considerados de maior relevância.

5.3. Próximas Etapas

Para melhorar o cenário de ensino identificado neste *survey*, pretende-se, como próxima etapa desta pesquisa, definir uma ementa que foque nos tópicos considerados relevantes para a formação profissional dos alunos de Engenharia de Software. Além disto, pretende-se definir uma abordagem de ensino mais voltada para o desenvolvimento de competências e habilidades profissionais nos alunos, a fim de aumentar a aprendizagem destes tópicos e, consequentemente, formar profissionais mais adequados para atender às demandas da indústria de software.

Tal melhoria exige um trabalho mais colaborativo entre pesquisadores e profissionais da indústria com o intuito de se concentrar no seguimento de práticas de capacitação que realmente desenvolvam competências e habilidades profissionais e reflitam a realidade do contexto no qual os alunos atuarão após a graduação. Acredita-se que a análise dos resultados deste *survey* é um passo para obter informações oportunas e valiosas, além de poder unir pesquisadores e profissionais da indústria de software em torno de um tema de interesse comum: a formação profissional de alunos.

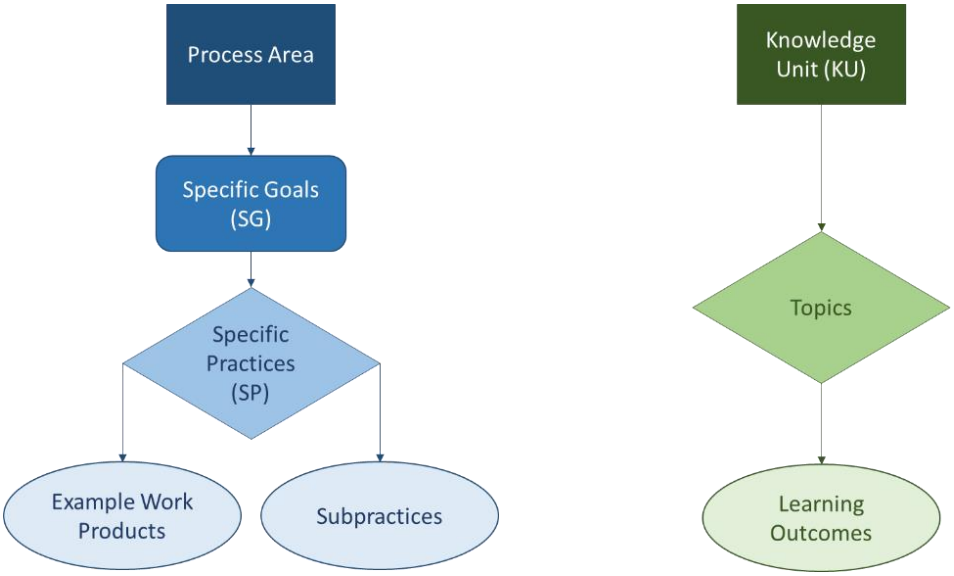
APÊNDICE B – MAPEAMENTO CURRÍCULOS X MODELOS

1. INTRODUÇÃO

O mapeamento apresentado neste documento objetiva relacionar os tópicos de Engenharia de Software (ES) do currículo de referência da ACM/IEEE (2013) com as práticas do modelo CMMI-DEV (SEI, 2010). Este mapeamento foi realizado por 3 profissionais da área de ES, que possuem experiência tanto da implementação do modelo CMMI-DEV em empresas de software quanto da aplicação do currículo da ACM/IEEE em disciplinas de ES.

A primeira etapa deste mapeamento ocorreu à nível de estrutura, conforme apresenta a Figura 1.

Figura 1 Mapeamento entre a estrutura do Modelo CMMI-DEV e do Currículo da ACM/IEEE



2. MAPEAMENTO

A segunda etapa consistiu no mapeamento à nível de conceitos, conforme apresenta a Tabela 1.

Tabela 1 Mapeamento entre os conceitos do Modelo CMMI-DEV e do Currículo da ACM/IEEE

Conceito do CMMI-DEV	Conceito da ACM/IEEE	Justificativa
Área de Processo (*Process Area)*	Unidade de Conhecimento (*Knowledge Unit*)	Uma área de processo é um conjunto de práticas relacionadas que, implementadas coletivamente, satisfazem um conjunto de objetivos considerados importantes para melhoria de uma área, como por exemplo Gerência de Requisitos (*Requirements Management* - REQM). De maneira similar, uma unidade de conhecimento é um conjunto de tópicos relacionados que compõem o conhecimento de uma área da ES, como por exemplo Engenharia de Requisitos (*Requirements Engineering*).
Objetivo Específico (*Specific Goals*)	<<Não identificado>>	Não foi identificado nenhum conceito no currículo de referência da ACM/IEEE que correspondesse a um objetivo específico.
Prática Específica (*Specific Practices*)	Tópicos (*Topics*)	Uma prática específica é uma descrição de uma atividade que é considerada importante para o atendimento de um objetivo específico associado, como por exemplo "SP 1.1 Levantar Necessidades". Já um tópico é a descrição de um conhecimento relacionado a uma unidade, como por exemplo "Elicitação de requisitos de software".
Subpráticas (*Subpractices*)	Aprendizagem Esperada (*Learning Outcomes*)	Por fim, definisse-se uma subprática é uma descrição detalhada que providencia orientação para interpretação e implementação de uma prática específica, como por exemplo "Estabelecer critérios objetivos para avaliação e aceitação de requisitos". Uma aprendizagem esperada está relacionada ao resultado do ensino de um tópico específico, como por exemplo "Realizar uma avaliação de um conjunto de requisitos de software para determinar a qualidade dos requisitos no que diz respeito às boas características de requisitos".

A terceira etapa consistiu no mapeamento entre as unidades de conhecimento da ACM/IEEE e as áreas de processo do CMMI-DEV, conforme apresenta a Tabela 2.

Tabela 2 Mapeamento entre as unidades de conhecimento da ACM/IEEE e as áreas de processo do CMMI-DEV

Unidade de Conhecimento da ACM/IEEE	**Áreas de Processo do CMMI-DEV**	**Categoria das Áreas de Processo do CMMI-DEV**
Engenharia de Requisitos (*Requirements Engineering*)	Gestão de Requisitos (REQM - *Requirements Management*)	Gestão de Projeto (*Project Management*)
	Desenvolvimento de Requisitos (RD - *Requirements Development*)	Engenharia (*Engineering*)
Processos de Software (*Software Processes*)	Definição dos Processos da Organização (OPD - *Organizational Process Definition*)	Gestão de Processo (*Process Management*)
	Treinamento na Organização (OT - *Organizational Training*)	
	Gestão de Performance Organizacional (OPM - *Organizational Performance Management*)	
	Desempenho dos Processos da Organização (OPP - Organizational Process Performance)	
	Foco nos Processos da Organização (OPF - *Organizational Process Focus*)	

Unidade de Conhecimento da ACM/IEEE	**Áreas de Processo do CMMI-DEV**	**Categoria das Áreas de Processo do CMMI-DEV**
Gerenciamento de Projeto de Software (*Software Project Management*)	Planejamento de Projeto (PP - *Project Planning*)	Gestão de Projeto (*Project Management*)
	Monitoramento e Controle de Projeto (PMC - *Project Monitoring and Control*)	
	Gestão de Requisitos (REQM - *Requirements Management*)	
	Gestão Integrada de Projeto (IPM - *Integrated Project Management*)	
	Gestão Quantitativa de Projeto (QPM - *Quantitative Project Management*)	
	Gestão de Riscos (RSKM - *Risk Management*)	
	Gestão de Contrato com Fornecedores (SAM - *Supplier Agreement Management*)	
Projeto de Software (*Software Design*)	Garantia da Qualidade de Processo e Produto (PPQA - *Process and Product Quality Assurance*)	Suporte (*Support*)
	Medição e Análise (MA - *Measurement and Analysis*)	
	Solução Técnica (TS - *Technical Solution*)	Engenharia (*Engineering*)

Unidade de Conhecimento da ACM/IEEE	**Áreas de Processo do CMMI-DEV**	**Categoria das Áreas de Processo do CMMI-DEV**
Verificação e Validação de Software (*Software Verification and Validation*)	Validação (VAL - *Validation*)	Engenharia (*Engineering*)
	Verificação (VER - *Verification*)	
Ferramentas e Ambientes (*Tools and Environments*)	Gestão de Configuração (CM - *Configuration Management*)	Suporte (*Support*)
	Medição e Análise (MA - *Measurement and Analysis*)	
	Validação (VAL - *Validation*)	Engenharia (*Engineering*)
	Verificação (VER - *Verification*)	
	Solução Técnica (TS - *Technical Solution*)	
	Desenvolvimento de Requisitos (RD - *Requirements Development*)	
	Integração de Produto (PI - *Product Integration*)	

A quarta etapa consistiu no mapeamento entre os tópicos da ACM/IEEE e as práticas específicas do CMMI-DEV, conforme apresenta, conforme apresenta a Tabela 3. Como cada prática específica está relacionada a um determinado nível de maturidade do modelo CMMI-DEV, inseriu-se uma coluna para representar esta informação. Adicionalmente, a fim de justificar a relação entre os tópicos e práticas, inseriu-se uma coluna com a justificativa do mapeamento.

Tabela 3 Mapeamento entre os tópicos da ACM/IEEE e as práticas específicas do CMMI-DEV

Unidade de Conhecimento	Tópicos	Práticas Específicas	Área de Processo	Nível	Justificativa
Engenharia de Requisitos	Descrição dos requisitos funcionais usando, por exemplo, casos de uso ou histórias do usuário	SP 3.2 Estabelecer uma Definição da Funcionalidade Requerida	Desenvolvimento de Requisitos (RD)	3	Esta SP consiste em descrever o que se pretende que o produto faça, ou seja, descrever seus requisitos funcionais. Estas funcionalidades podem incluir ações, sequências, entradas, saídas ou outras informações que explicam a maneira que o produto será utilizado.
	Propriedades de requisitos, incluindo a consistência, validade, integridade e viabilidade	SP 3.3 Analisar Requisitos			Esta SP consiste em analisar os requisitos para assegurar que são necessários, implementáveis e factíveis (viabilidade), completos (integridade), verificáveis (validade) e suficientes (consistência).
	Elicitação de requisitos de software	SP 1.1 Levantar Necessidades			Esta SP consiste em elicitar os requisitos do software com as partes interessadas, a partir de suas expectativas, restrições e interfaces para todas as fases do ciclo de vida do produto. Além disso, envolve a identificação proativa de requisitos adicionais não fornecidos explicitamente pelos clientes.
	Descrição dos dados do sistema utilizando, por exemplo, diagramas de classe ou	SP 2.1 Estabelecer Requisitos de Produto e de Componente de Produto			Esta SP consiste em estabelecer os requisitos de produto e de componente de produto, com base nos requisitos de cliente. Estes requisitos de produto são a expressão dos requisitos dos clientes em termos técnicos, tal qual

Unidade de Conhecimento	Tópicos	Práticas Específicas	Área de Processo	Nível	Justificativa
	diagramas entidade-relacionamento				ocorre com a descrição técnica dos dados dos diagramas de classe ou diagramas entidade-relacionamento.
	Requisitos não-funcionais e sua relação com a qualidade de software	SP 3.3 Analisar Requisitos			Esta SP consiste em analisar os conceitos operacionais e cenários para descobrir novos requisitos, como por exemplos, os não-funcionais. Além disso, consiste em identificar a relação destes requisitos com o desempenho do software através de medidas que serão acompanhadas durante o desenvolvimento.
	Análise de requisitos (técnicas de modelagem)	SP 3.4 Analisar Requisitos Visando ao Balanceamento			Esta SP consiste em analisar requisitos através de modelos, simulações e prototipação a fim de balancear as necessidades e restrições das partes interessadas.
	Avaliação e utilização de especificações de requisitos	SP 3.5 Validar Requisitos			Esta SP consiste em avaliar a adequação e a completude dos requisitos por meio da utilização de representações do produto (por exemplo: protótipos, simulações, modelos, cenários e storyboards) e da obtenção de feedback das partes interessadas relevantes a respeito dessas representações.
	Prototipagem	SP 3.4 Analisar Requisitos Visando ao Balanceamento			Esta SP consiste em analisar requisitos através de prototipação a fim de contemplar os requisitos das partes interessadas.

Unidade de Conhecimento	Tópicos	Práticas Específicas	Área de Processo	Nível	Justificativa
	Especificação de requisitos	SP 1.2 Desenvolver Requisitos de Cliente			Esta SP consiste em traduzir as necessidades, expectativas, restrições e interfaces das partes interessadas em requisitos de cliente documentados.
	Validação de requisitos	SP 3.5 Validar Requisitos			Esta SP consiste em identificar questões críticas de validação e explicitar necessidades e requisitos do cliente não declarados.
	Rastreamento de requisitos	SP 1.4 Manter Rastreabilidade Bidirecional dos Requisitos	Gestão de Requisitos (REQM)	2	Esta SP consiste em manter a rastreabilidade de um requisito com seus requisitos associados e com seus produtos de trabalho relacionados.
Processos de Software	Introdução aos modelos de processo de software (por exemplo, cascata, incremental, ágil)	SP 1.2 Estabelecer Descrições de Modelos de Ciclo de Vida	Definição dos Processos da Organização (OPD)	3	Esta SP consiste em elaborar modelos de ciclo de vida adequados para diferentes clientes ou situações, por exemplo cascata, incremental, iterativo. Estes modelos de ciclo de vida são utilizados frequentemente para definir as fases do projeto.
	Avaliação de modelo de processo de software	SP 1.3 Estabelecer Critérios e Diretrizes para Adaptação			Esta SP consiste em avaliar os modelos de processos, dentre os aprovados pela organização, a fim de adaptá-los para uma nova linha de produto ou novo ambiente de trabalho.
	Melhoria de processo	SP 1.3 Identificar Melhorias para os	Foco nos Processos da Organização (OPF)	3	Esta SP consiste em identificar melhorias para os processos e ativos de processo da organização.

Unidade de Conhecimento	Tópicos	Práticas Específicas	Área de Processo	Nível	Justificativa
		Processos da Organização			
	Medição de processo de software	SP 1.2 Estabelecer Medidas de Desempenho de Processo	Desempenho dos Processos da Organização (OPP)	4	Esta SP consiste em estabelecer e manter medidas a serem incluídas nas análises de desempenho de processo da organização.
	Conceitos de qualidade de software	SP 1.3 Estabelecer Objetivos para Qualidade e para Desempenho de Processo			Esta SP consiste em definir objetivos quantitativos para a qualidade e para o desempenho de processo da organização. Neste processo, se faz necessário revisar os conceitos da organização relativos à qualidade e ao desempenho de processo.
	Modelo de maturidade e capacidade de processo de software	<<Contemplado>>	-	-	Os próprios modelos de qualidade adotados neste mapeamento (CMMI-DEV e MR-MPS-SW) atendem a este tópico.
Gerenciamento de Projeto de Software	Participação da equipe (responsabilidades, reuniões, agenda de trabalho, resolução de conflitos, etc)	SP 2.4 Planejar Recursos do Projeto	Planejamento de Projeto (PP)	2	Esta SP consiste em determinar requisitos para composição da equipe.
		SP 2.5 Planejar Habilidades e Conhecimento Necessários			Esta SP consiste em planejar as habilidades e conhecimento necessários para apoiar a execução do projeto.

Unidade de Conhecimento	Tópicos	Práticas Específicas	Área de Processo	Nível	Justificativa
		SP 2.6 Planejar o Envolvimento das Partes Interessadas			Esta SP consiste em identificar as pessoas e funções que precisam ter representação no projeto, descrevendo sua relevância e grau de interação em atividades específicas do projeto.
	Estimativa de esforço	SP 1.2 Estabelecer Estimativas para Atributos de Produtos de Trabalho e de Tarefas.			Esta SP consiste em estabelecer e manter estimativas de esforço, custo e prazo.
	Técnicas de medição e estimativa de software	SP 1.2 Estabelecer Estimativas para Atributos de Produtos de Trabalho e de Tarefas.			Esta SP consiste em estabelecer e manter estimativas de esforço, custo e prazo através de técnicas bem sucedidas.
	Gerenciamento de Projeto (planejamento e acompanhamento, ferramentas de gerenciamento de projeto e análise de custo/benefício)	SP 2.7 Estabelecer o Plano do Projeto			Esta SP consiste em definir e manter um plano global do projeto que agrupe de maneira lógica: considerações sobre o ciclo de vida do projeto; tarefas técnicas e de gestão; orçamentos e cronogramas; marcos; requisitos para gestão de dados, identificação de riscos, recursos e habilidades; e identificação de partes interessadas e suas interações.
		SP 1.1 Monitorar os Parâmetros de Planejamento do Projeto	Monitoramento e Controle de Projeto (PMC)	2	Esta SP consiste em monitorar os valores reais dos parâmetros de planejamento de projeto em relação ao plano de projeto.

Unidade de Conhecimento	Tópicos	Práticas Específicas	Área de Processo	Nível	Justificativa
		SP 1.4 Gerenciar o Desempenho do Projeto	Gestão Quantitativa de Projeto (QPM)	4	Esta SP consiste em monitorar o projeto para determinar se os objetivos para qualidade e para desempenho de processo no projeto serão satisfeitos. Desta forma, é possível fazer uma análise de custo/benefício do projeto.
	Riscos (identificação e gestão, análise e avaliação, tolerância e planejamento)	SP 2.1 Identificar Riscos	Gestão de Riscos (RSKM)	3	Esta SP consiste em identificar e documentar os riscos.
		SP 2.2 Avaliar, Categorizar e Priorizar Riscos			Esta SP consiste em analisar e avaliar os riscos identificados.
		SP 3.1 Elaborar Planos de Mitigação de Riscos			Esta SP consiste em elaborar um plano geral de mitigação de riscos para que o projeto possa coordenar a implementação de planos individuais de mitigação e de contingência.
Projeto de Software	Princípios de projeto de sistema: níveis de abstração (projeto arquitetônico e projeto detalhado), separação de interesses, a ocultação de informações, o acoplamento e	SP 2.1 Desenvolver o Design do Produto ou dos Componentes de Produto	Solução Técnica (TS)	3	Esta SP consiste em desenvolver projetos de sistemas preliminares e detalhados. O projeto preliminar estabelece as principais funcionalidades e características do produto e sua arquitetura, incluindo a identificação de componentes de produto, estados do sistema, principais interfaces entre componentes e interfaces externas do produto. Já o projeto detalhado define completamente a estrutura e a funcionalidade dos componentes do produto.

Unidade de Conhecimento	Tópicos	Práticas Específicas	Área de Processo	Nível	Justificativa
	coesão, reuso de estruturas padrão				
	Paradigmas de projeto, tais como projeto estruturado, orientada a objetos, orientado a evento, a nível de componente, orientada a aspecto, orientada a função, orientada a serviços				Esta SP consiste em definir arquiteturas que podem incluir padrões e regras de design que regulamentam o desenvolvimento de componentes de produto e suas interfaces, assim como orientações para auxiliar os desenvolvedores de produto. Desta forma, pode-se definir uma arquitetura aderente a determinado paradigma de projeto, tal como orientada a objetos, orientado a evento, orientada a aspecto, orientada a serviços, etc.
	Modelos estruturais e comportamentais de projetos de software				Esta SP consiste em definir modelos estruturais e mecanismos comportamentais que ou satisfazem diretamente ou dão suporte à satisfação dos requisitos. Estas estruturas e mecanismos são definidas através de arquiteturas.
	Padrões de projeto				Esta SP consiste em identificar e elaborar padrões de projeto que regulamentam o desenvolvimento de componentes de software e auxiliam os desenvolvedores. Além disto, deve-se assegurar que o projeto seja aderente a estes padrões.
	Conceitos de arquitetura de				Esta SP consiste em desenvolver um design para o produto, que abrange padrões arquiteturais. Neste

Unidade de Conhecimento	Tópicos	Práticas Específicas	Área de Processo	Nível	Justificativa
	software e padrões arquiteturais (por exemplo, cliente-servidor)				processo, várias arquiteturas que dão suporte às soluções alternativas podem ser desenvolvidas e analisadas para determinar suas vantagens e desvantagens com relação aos requisitos de arquitetura.
Verificação e Validação de Software	Conceitos de verificação e validação	SP 1.1 Selecionar Produtos de Trabalho para Verificação	Verificação (VER)	3	Esta SP consiste em identificar os métodos de verificação que estão disponíveis para uso. Neste processo, obtêm-se os conceitos relacionados à verificação.
		SP 1.1 Selecionar Produtos de Trabalho para Validação	Validação (VAL)		Esta SP consiste em identificar as principais questões associadas à validação do produto, tais como princípios, características e fases.
	Inspeções, revisões e auditorias	SP 1.3 Estabelecer Procedimentos e Critérios de Verificação	Verificação (VER)		Esta SP consiste em definir procedimentos de verificação, como inspeções, revisões e auditorias de código-fonte.
		SP 1.3 Estabelecer Procedimentos e Critérios de Validação	Validação (VAL)		Esta SP consiste em definir procedimentos de validação, como inspeções, revisões e auditorias em produtos de trabalho.

Unidade de Conhecimento	Tópicos	Práticas Específicas	Área de Processo	Nível	Justificativa
	Tipos de testes, incluindo interface homem-computador, usabilidade, confiabilidade, segurança, conformidade com especificação	SP 1.3 Estabelecer Procedimentos e Critérios de Verificação	Verificação (VER)		Esta SP consiste em definir procedimentos de verificação, como testes de interface, usabilidade, confiabilidade, segurança e conformidade que serão aplicados no software desenvolvido.
		SP 1.3 Estabelecer Procedimentos e Critérios de Validação	Validação (VAL)		Esta SP consiste em definir procedimentos de verificação, como análise da conformidade dos produtos de trabalhos com os requisitos do software.
	Fundamentos de teste (unidade, integração, validação e testes de sistema, criação do plano de teste e casos de teste, caixa-preta e caixa branca, teste de regressão, automação de teste, controle de defeitos)	SP 1.2 Estabelecer Ambiente de Verificação	Verificação (VER)		Esta SP consiste em identificar os procedimentos de verificação que estão disponíveis para uso e adaptação, como os principais tipos e fundamentos de testes (unidade, integração, regressão, caixa-preta e caixa branca, etc).
Ferramentas e Ambientes	Gerência de configuração e controle de versão	SP 1.2 Estabelecer um Sistema de Gestão de Configuração	Gestão de Configuração (CM)	2	Esta SP consiste em estabelecer um sistema de gerência da configuração que seja capaz de realizar controle de versão. A fim de atender esta SP, o ideal é utilizar uma ferramenta que automatize esse processo.

Unidade de Conhecimento	Tópicos	Práticas Específicas	Área de Processo	Nível	Justificativa
	Análise de requisitos e ferramentas de modelagem	SP 3.4 Analisar Requisitos Visando ao Balanceamento	Desenvolvimento de Requisitos (RD)	3	Esta SP consiste em analisar requisitos através de modelos. Logo, seria relevante o apoio de uma ferramenta de modelagem.

A quinta e última etapa consistiu no mapeamento entre as aprendizagens esperadas do currículo de referência da ACM/IEEE e as subpráticas do CMMI-DEV, conforme apresenta a Tabela 4.

Tabela 3 Mapeamento entre as aprendizagens da ACM/IEEE e as subpráticas do CMMI-DEV

Unidade de Conhecimento	Aprendizagem Esperada	Subprática	Área de Processo
Engenharia de Requisitos	1. Listar os principais componentes de um caso de uso ou descrição semelhante de algum comportamento que é necessário para um sistema.	SP 3.2 Subprática 1. Analisar e quantificar as funcionalidades requeridas pelos usuários finais.	Desenvolvimento de Requisitos (RD)
	2. Descrever como o processo de engenharia de requisitos apoia a elicitação e validação de requisitos comportamentais.	SP 3.5 - Subprática 2. Explorar a adequação e completude dos requisitos por meio do desenvolvimento de representações do produto e da obtenção de feedback das partes interessadas relevantes a respeito dessas representações.	
	3. Interpretar um determinado modelo de requisitos para um sistema de software simples.	SP 3.4 - Subprática 1. Utilizar modelos, simulações e prototipação de uso comprovado para analisar o balanceamento entre necessidades e restrições das partes interessadas.	

Unidade de Conhecimento	Aprendizagem Esperada	Subprática	Área de Processo
	4. Descrever os desafios fundamentais de técnicas comuns usadas para elicitação de requisitos.	SP 1.1 - Subprática 1. Envolver as partes interessadas relevantes utilizando métodos para levantamento de necessidades, expectativas, restrições e interfaces externas.	
	5. Listar os principais componentes de um modelo de dados (por exemplo, diagramas de classe ou diagramas ER).	SP 2.1 - Subprática 1. Desenvolver os requi sitos em termos técnicos adequados para a realização do design de produto e de componente de produto.	
	6. Identificar os requisitos funcionais e não-funcionais numa determinada especificação de requisitos para um sistema de software.	SP 3.3 - Subprática 1. Analisar necessidades, expectativas, restrições e interfaces externas das partes interessadas para organizá-los em assuntos relacionados e solucionar conflitos.	
	7. Realizar uma avaliação de um conjunto de requisitos de software para determinar a qualidade dos requisitos no que diz respeito às boas características de requisitos.	SP 3.3 - Subprática 3. Analisar requisitos para assegurar que eles sejam completos, factíveis, implementáveis e verificáveis.	
	8. Aplicar elementos-chave e métodos comuns para elicitação e análise para produzir um conjunto de requisitos para um sistema de software de médio porte.	"SP 3.5 - Subprática 3. Avaliar o design à medida que ele avança no contexto do ambiente de validação dos requisitos para identificar questões críticas de validação e explicitar necessidades e requisitos do cliente não declarados."	

Unidade de Conhecimento	Aprendizagem Esperada	Subprática	Área de Processo
	11. Traduzir em linguagem natural uma especificação de requisitos de software (por exemplo, um contrato de componente de software) escrito em uma linguagem de especificação formal.	SP 1.2 - Subprática 1. Traduzir as necessidades, expectativas, restrições e interfaces das partes interessadas em requisitos de cliente documentados.	
	12. Criar um protótipo de um sistema de software para limitar os riscos nos requisitos.	SP 3.4 - Subprática 2. Realizar uma análise dos riscos relacionados aos requisitos e à arquitetura funcional.	
	13. Diferenciar entre rastreabilidade a horizontal e vertical e explicar as suas funções no processo de validação de requisitos.	"SP 1.4 - Subprática 2. Manter a rastreabilidade de um requisito com seus requisitos detalhados e com sua alocação a funções, interfaces, pessoas, processos e produtos de trabalho"	Gestão de Requisitos (REQM)
Processos de Software	2. Descrever as vantagens e desvantagens relativas entre os vários modelos de processos.	SP 1.3 - Subprática 1. Especificar os critérios de seleção e procedimentos para adaptação do conjunto de processos-padrão da organização.	Definição dos Processos da Organização (OPD)
	7. Comparar vários modelos de processo no que diz respeito ao seu valor para o desenvolvimento de determinadas classes de sistemas de software, tendo em conta questões como a estabilidade exigência, tamanho e características não-funcionais.	SP 1.2 - Subprática 1. Selecionar os modelos de ciclo de vida com base nas necessidades dos projetos e da organização.	

Unidade de Conhecimento	Aprendizagem Esperada	Subprática	Área de Processo
	8. Definir a qualidade do software e descrever o papel das atividades de garantia da qualidade no processo de software.	SP 1.3 - Subprática 3. Definir a prioridade dos objetivos para qualidade e para desempenho de processo da organização.	Desempenho dos Processos da Organização (OPP)
	9. Descrever a intenção e semelhanças fundamentais entre as abordagens de melhoria de processos.	SP 1.3 - Subprática 3. Identificar e documentar as melhorias de processo a serem implementadas.	Foco nos Processos da Organização (OPF)
	12. Explicar o papel de modelos de maturidade em processos de melhoria de processos.	<<Contemplado>>	Todas
	13. Descrever várias métricas de processo para avaliar e controlar um projeto.	SP 1.2 - Subprática 2. Selecionar medidas para fornecer visibilidade apropriada sobre a qualidade e o desempenho de processo da organização.	Desempenho dos Processos da Organização (OPP)
Gerenciamento de Projetos de Software	3. Identificar e justificar papéis necessários em uma equipe de desenvolvimento de software.	SP 2.5 - Subprática 1. Identificar habilidades e conhecimento necessários para a execução do projeto.	Planejamento de Projeto (PP)
	5. Aplicar uma estratégia de resolução de conflitos em um ambiente de equipe.	SP 2.4 - Subprática 2. Determinar requisitos para composição da equipe.	
	6. Utilizar um método ad hoc para estimar esforço de desenvolvimento de software (por	SP 1.2 - Subprática 2. Utilizar métodos apropriados para determinar os atributos de produtos de trabalho	

Unidade de Conhecimento	Aprendizagem Esperada	Subprática	Área de Processo
	exemplo, tempo) e comparar com o esforço real necessário.	e de tarefas que serão utilizados para estimar os requisitos de recursos.	
		SP 1.2 - Subprática 3. Estimar os atributos de produtos de trabalho e de tarefas.	
	7. Listar vários exemplos de riscos de software.	SP 2.1 - Subprática 1. Identificar os riscos associados a custo, prazo e desempenho.	Gestão de Riscos (RSKM)
	8. Descrever o impacto do risco em um ciclo de desenvolvimento de software.	SP 2.2 - Subprática 1. Avaliar os riscos identificados utilizando os parâmetros definidos para riscos.	
	15. Acompanhar o progresso de algum estágio de um projeto usando métricas de projeto adequadas.	SP 1.4 - Subprática 1. Revisar periodicamente o desempenho de cada subprocesso e a capacidade de cada subprocesso selecionado para ser gerenciado estatisticamente, a fim de avaliar o progresso na direção dos objetivos para qualidade e para desempenho de processo no projeto.	Gestão Quantitativa de Projeto (QPM)
	17. Usar uma ferramenta de gerenciamento de projetos para auxiliar na atribuição e acompanhamento de tarefas em um projeto de desenvolvimento de software.	SP 1.1 - Subprática 1. Monitorar o progresso em relação ao cronograma.	Monitoramento e Controle de Projeto (PMC)
	19. Identificar os riscos e descrever abordagens para a gestão dos riscos (prevenção, a aceitação, a transferência, mitigação), e caracterizar os pontos fortes e fracos de cada um.	SP 3.1 - Subprática 1. Determinar os níveis e limiares que definem quando um risco torna-se inaceitável e quando a execução de um plano de mitigação ou de contingência deve ser disparada.	Gestão de Riscos (RSKM)

Unidade de Conhecimento	Aprendizagem Esperada	Subprática	Área de Processo
Projeto de Software	2. Usar um paradigma de projeto para projetar um sistema de software simples, e explicar como princípios de design foram aplicadas neste projeto.	SP 2.1 - Subprática 3. Assegurar que o design seja aderente a padrões e critérios de design aplicáveis.	Solução Técnica (TS)
	4. Dentro do contexto de um único paradigma de projeto, descrever um ou mais padrões de design que também possam ser aplicados para a concepção de um sistema de software simples.	SP 2.1 - Subprática 2. Identificar, elaborar ou obter métodos apropriados de design para o produto.	
	6. Criar modelos adequados para a estrutura e o comportamento de produtos de software a partir das suas especificações de requisitos.	SP 2.1 - Subprática 2. Identificar, elaborar ou obter métodos apropriados de design para o produto.	
	8. Para o projeto de um simples sistema de software dentro do contexto de um único paradigma de projeto descrever a arquitetura de software de sistema.	SP 2.1 - Subprática 2. Identificar, elaborar ou obter métodos apropriados de design para o produto.	
	16. Discutir e selecionar a arquitetura de software apropriado para um sistema simples adequado para um determinado cenário.	SP 2.1 - Subprática 2. Identificar, elaborar ou obter métodos apropriados de design para o produto.	
Verificação e Validação de Software	1. Distinguir entre a validação e verificação do programa.	SP 1.1 - Subprática 3. Identificar os métodos de verificação que estão disponíveis para uso.	Verificação (VER)

Unidade de Conhecimento	Aprendizagem Esperada	Subprática	Área de Processo
		SP 1.1 - Subprática 1. Identificar ao longo da vida do projeto as principais questões associadas à validação do produto ou componente de produto, tais como princípios, características e fases.	Validação (VAL)
	3. Realizar, como parte de uma atividade de equipe, uma inspeção a um segmento de código de tamanho médio.	SP 2.2 - Subprática 2. Identificar e documentar defeitos e outras questões críticas no produto de trabalho.	Verificação (VER)
		SP 1.3 - Subprática 1. Revisar os requisitos de produto para assegurar que questões críticas que afetam a validação do produto ou componente de produto sejam identificadas e resolvidas.	Validação (VAL)
	4. Descrever e distinguir entre os diferentes tipos e níveis de testes (unidade, integração, sistemas e aceitação).	SP 1.3 - Subprática 4. Identificar todos os equipamentos e componentes de ambiente necessários para dar suporte à verificação.	Verificação (VER)
		SP 1.3 - Subprática 2. Documentar ambiente, cenário operacional, procedimentos, entradas, saídas e critérios para a validação dos produtos ou componentes de produto selecionados.	Validação (VAL)
Ferramentas e Ambiente	2. Descrever como controle de versão pode ser usado para ajudar a controlar o gerenciamento de versão de software.	SP 1.2 - Subprática 1. Estabelecer um mecanismo para gerenciar vários níveis de controle de gestão de configuração.	Gestão de Configuração (CM)

Unidade de Conhecimento	Aprendizagem Esperada	Subprática	Área de Processo
	5. Descrever as questões que são importantes na escolha de um conjunto de ferramentas para o desenvolvimento de um sistema de software particular, incluindo ferramentas para rastreamento de requisitos, modelagem de projeto, implementação, construção automática e testes.	SP 3.4 - Subprática 1. Utilizar modelos, simulações e prototipação de uso comprovado para analisar o balanceamento entre necessidades e restrições das partes interessadas.	Desenvolvimento de Requisitos (RD)

APÊNDICE C – LEVANTAMENTO PRÁTICAS DA INDÚSTRIA

1. INTRODUÇÃO

Na área de ensino de Engenharia de Software (ES), considera-se importante não só teorizar sobre os tópicos da área, mas colocá-lo em prática. Neste sentido, os autores desta pesquisa de doutorado consideram possível adequar determinadas práticas da indústria para o contexto de uma disciplina da área de ES. No entanto, há escassez de professores qualificados na academia, que tenham de fato atuado na indústria.

A fim de atender esta demanda, tanto de identificar práticas de capacitação da indústria e estratégias de avaliação, realizou-se um levantamento com consultores em Melhoria do Processo de Software (MPS). O objetivo deste levantamento, disponível no *Google Forms* (*https://goo.gl/FSfsiQ*), foi identificar junto à especialistas da área (consultores de MPS e implementadores de modelos de qualidade que também atuem como professores de graduação) sobre quais as estratégias de capacitação adotadas em programas de MPS.

A Tabela 1 mostra as perguntas feitas aos consultores neste levantamento.

Tabela 1 – Perguntas do Levantamento de Práticas de Capacitação da Indústria

Pergunta	**Respostas Identificadas**
Quais modelos de melhoria do processo de software implementa?	( ) MR-MPS-SW ( ) CMMI-DEV ( ) Outros
Qual a estratégia de capacitação adotada para os processos relacionados à Engenharia de Software?	Para cada unidade de conhecimento da ES: ( ) *Workshop* ( ) Dinâmica ( ) *Mentoring* ( ) *Coaching* ( ) Outros
Como o aprendizado é avaliado nestas estratégias?	( ) Prova Objetiva ( ) Prova Subjetiva ( ) Participação ( ) Frequência ( ) Outros

Para estas questões, consideram-se as seguintes definições:

- *Workshop*: seminário ou curso intensivo, de curta duração, em que técnicas e habilidades são demonstrados e aplicados;
- Dinâmica: instrumento que está dentro de um processo de formação, que possibilita a criação e recriação do conhecimento;
- *Mentoring*: um profissional mais experiente orienta e compartilha com profissionais mais jovens, experiências e conhecimentos;
- *Coaching*: consiste no desenvolvimento de competências e habilidades a partir de treinamento em técnicas específicas.

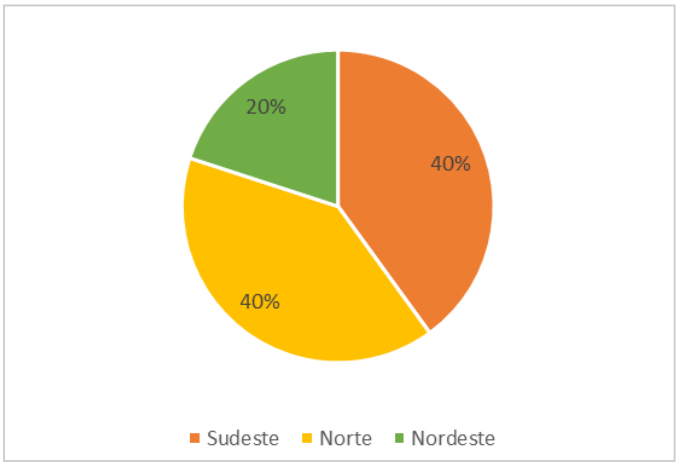
Além desta seção introdutória, a Seção 2 descreve a aplicação dos formulários do levantamento. As análises e discussões dos resultados obtidos neste levantamento são apresentados, respectivamente, nas Seções 3 e 4. Por fim, as limitações, as conclusões e as próximas etapas deste trabalho são apresentadas na Seção 5.

2. REALIZAÇÃO DO LEVANTAMENTO

Os dados foram coletados durante o período de 21 de Julho à 3 de Setembro de 2015. Durante esse período, recebeu-se respostas completas de 10 consultores. É uma baixa amostra, considerando que o mesmo foi enviado para cerca de 40 consultores.

Desta forma, a fim de reduzir este viés da amostragem, realizou-se o levantamento nas regiões norte, nordeste e sudeste do Brasil. Obtiveram-se respostas de 5 instituições implementadoras de MPS. A instituição implementadora de MPS que teve o maior número de participantes foi a SWQuality (Recife-PE), da qual participaram 5 consultores. Quanto a região das instituições onde esses consultores atuam como professores, Norte e Sudeste tiveram 40% de participantes cada uma e 20% na região Nordeste, conforme apresenta o Gráfico 1.

Gráfico 1 – Regiões das Instituições de Ensino dos Professores/Consultores

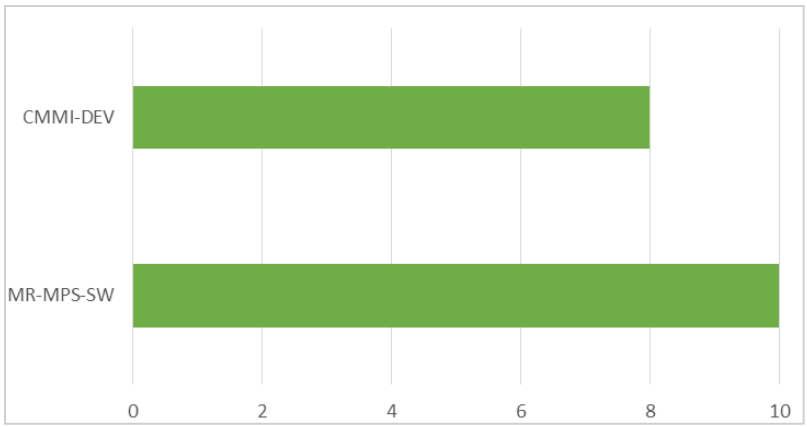


A média de tempo que estes consultores atuam como professores é de 13 anos e meio, sendo o menor tempo de docência 6 meses e o maior 32 anos. Já a média de tempo

que estes consultores atuam como implementadores de MPS é de 11 anos, sendo o menor tempo de consultoria 4 anos e o maior 32 anos.

Quanto aos modelos de MPS, 100% dos consultores já implementaram o Modelo de Referência MPS para Software (MR-MPS-SW) e 80% já implementaram o *Capability Maturity Model Integration for Development* (CMMI-DEV), conforme Gráfico 2.

Gráfico 2 – Modelos de MPS adotados por Professores/Consultores



A seguir, na Seção 3, são apresentados os resultados obtidos no levantamento no que diz respeitos às práticas de capacitação e estratégias de avaliação adotadas por estes consultores/professores em iniciativas de MPS.

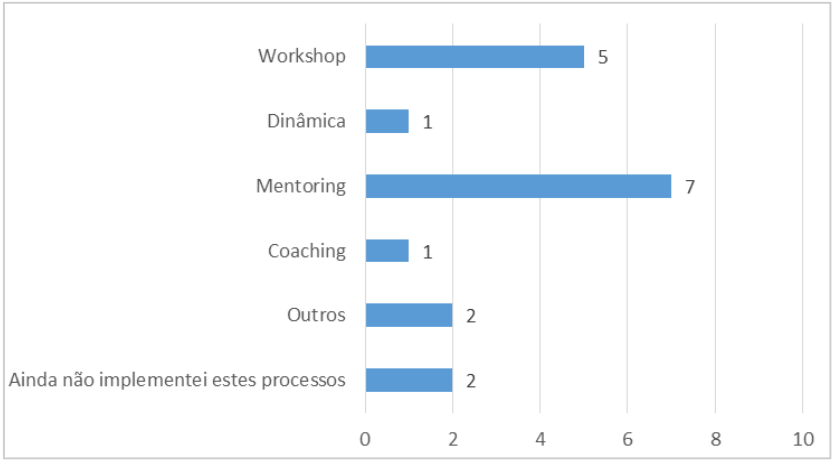
3. ANÁLISE DE DADOS

3.1. Práticas de Capacitação

A) Projeto de Software

Os processos relacionados à unidade de conhecimento Projeto de Software são: MR-MPS-SW (PCP - Projeto e Construção do Produto) e CMMI-DEV (TS - *Technical Solution*). Para esta unidade, foram adotadas as práticas de capacitação mostradas no Gráfico 3.

Gráfico 3 – Práticas de Capacitação para Projeto de Software

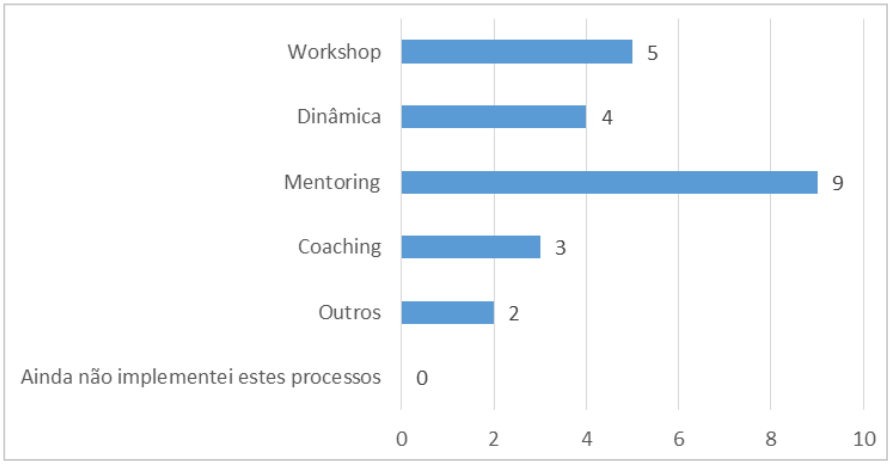


Observa-se que a maioria dos consultores realizam *workshops* e *mentoring* para capacitação de profissionais nesta unidade. Além destas, foram citadas outras práticas, como Cursos e Palestras. 2 consultores entrevistados ainda não implementaram os processos relacionados a esta unidade de conhecimento.

B) Gerenciamento de Projetos

Os processos relacionados à unidade de conhecimento Gerenciamento de Projetos são: MR-MPS-SW (GPR - Gerência de Projetos e GRI - Gerência de Riscos) e CMMI (PP - Project Planning, PMC - *Project Monitoring and Control*, QPM - *Quantitative Project Management* e RSKM - *Risk Management*). Para esta unidade, foram adotadas as práticas de capacitação mostradas no Gráfico 4.

Gráfico 4 – Práticas de Capacitação para Gerenciamento de Projetos



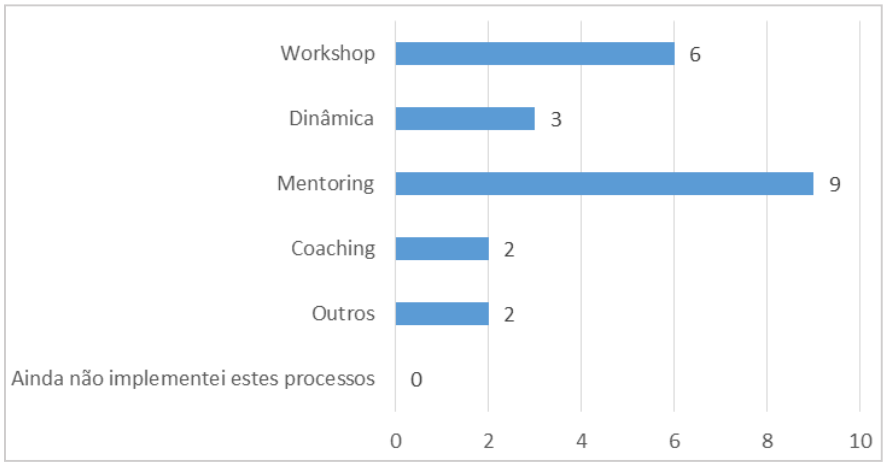
Observa-se que a grande maioria dos consultores realizam *mentoring*, *workshops* e dinâmicas para capacitação de profissionais nesta unidade. Além destas, foram citados Cursos e Palestras como outras práticas.

C) Processos de Software

Os processos relacionados à unidade de conhecimento Processos de Software são: MR-MPS-SW (DFP - Definição do Processo Organizacional e AMP - Avaliação e Melhoria do Processo Organizacional) e CMMI-DEV (OPD - *Organizational Process Definition*, OPP - *Organizational Process Performance* e OPF - *Organizational Process Focus*). Para esta unidade, foram adotadas as práticas de capacitação mostradas no Gráfico 5.

Observa-se que a maioria dos consultores realizam *mentoring* e *workshops* para capacitação de profissionais nesta unidade. Além destas, foram citadas outras práticas, como Cursos e Palestras.

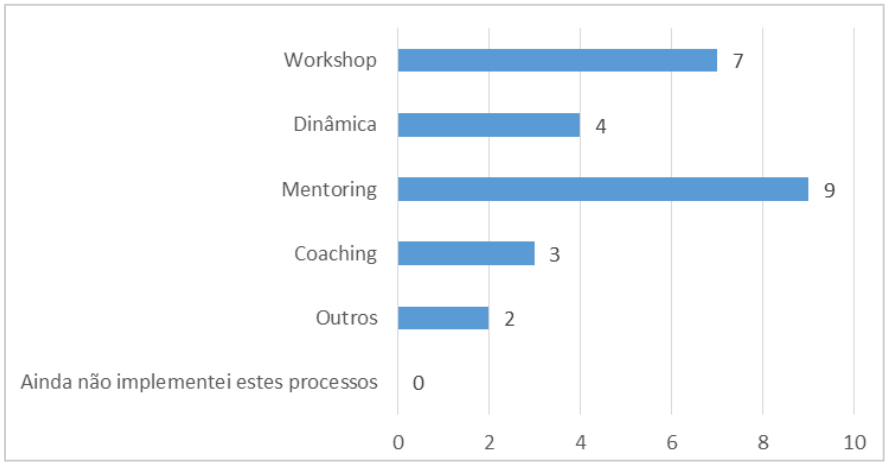
Gráfico 5 – Práticas de Capacitação para Processos de Software



D) Engenharia de Requisitos

Os processos relacionados à unidade de conhecimento Engenharia de Requisitos são: MR-MPS-SW (GRE - Gerência de Requisitos e DRE - Desenvolvimento de Requisitos) e CMMI-DEV (REQM - *Requirements Management* e RD - *Requirements Development*). Para esta unidade, foram adotadas as práticas de capacitação mostradas no Gráfico 6.

Gráfico 6 – Práticas de Capacitação para Engenharia de Requisitos

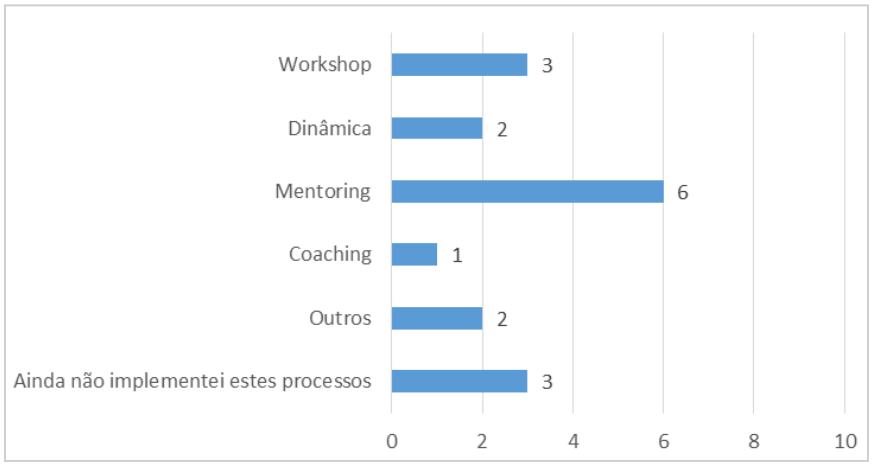


Observa-se que a grande maioria dos consultores realizam *mentoring*, *workshops* e dinâmicas para capacitação de profissionais nesta unidade. Além destas, foram citados Cursos e Palestras como outras práticas.

E) Verificação e Validação

Os processos relacionados à unidade de conhecimento Verificação e Validação são: MR-MPS-SW (VER - Verificação e VAL - Validação) e CMMI-DEV (VER - *Verification* e VAL - *Validation*). Para esta unidade, foram adotadas as práticas de capacitação mostradas no Gráfico 7.

Gráfico 7 – Práticas de Capacitação para Verificação e Validação

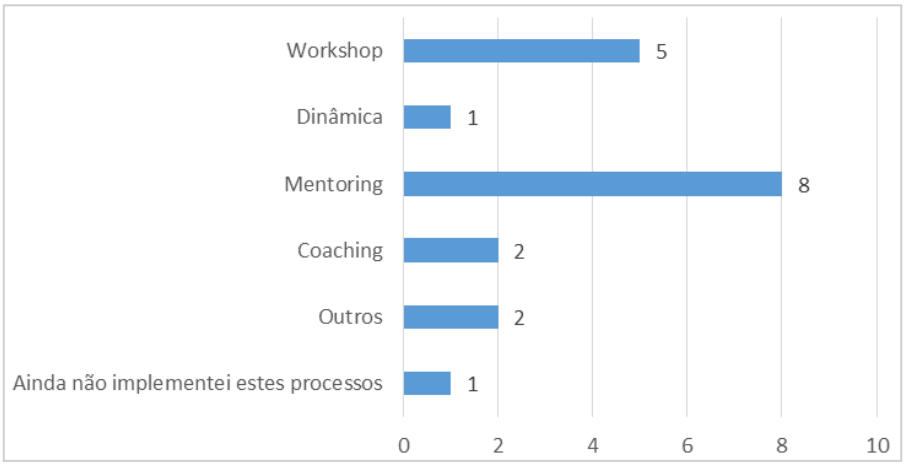


Observa-se que a maioria dos consultores realizam *mentoring* e *workshops* para capacitação de profissionais nesta unidade. Além destas, foram citadas outras práticas, como Cursos e Palestras. 3 consultores entrevistados ainda não implementaram estes processos.

F) Ambientes e Ferramentas

Os processos relacionados à unidade de conhecimento Ambientes e Ferramentas são: MR-MPS-SW (GCO - Gerência de Configuração e DRE - Desenvolvimento de Requisitos) e CMMI-DEV (CM - *Configuration Management* e RD - *Requirements Development*). Para esta unidade, foram adotadas as práticas de capacitação mostradas no Gráfico 8.

Gráfico 8 – Práticas de Capacitação para Ambientes e Ferramentas

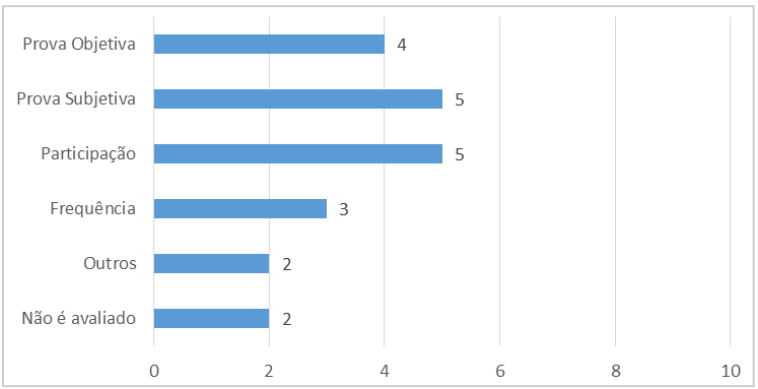


Observa-se que a grande maioria dos consultores realizam *mentoring* e *workshops* para capacitação de profissionais nesta unidade. Além destas, foram citados Cursos e Palestras como outras práticas. 1 consultor ainda não implementou os processos desta área.

3.2. Estratégias de Avaliação

Após listarem as práticas de capacitação adotadas na implementação de processos, os consultores listaram as estratégias de avaliação usadas neste processo. As estratégias identificadas são apresentadas no Gráfico 9.

Gráfico 9 – Estratégias de Avaliação adotadas por Consultores



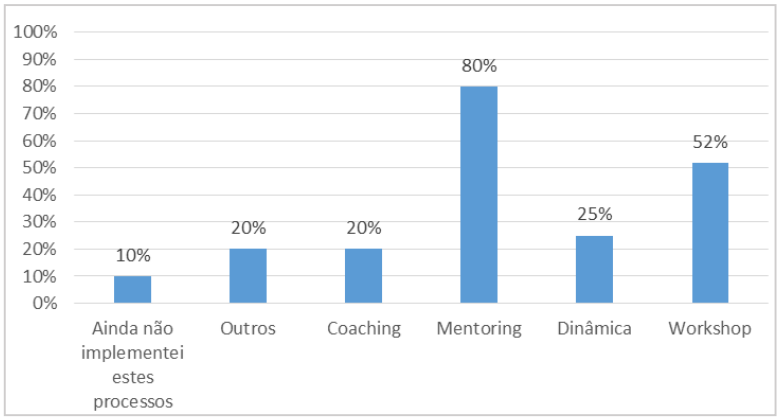
Observa-se que os consultores adotam principalmente provas subjetivas e a participação para avaliar o aprendizado. Além disso, realizam provas objetivas e consideram a frequência a fim de avaliar o comprometimento. Os consultores também citaram que avaliaram os profissionais através da observação da realização de atividades práticas em um projeto.

4. DISCUSSÕES

4.1. Sobre as Práticas de Capacitação

A partir da análise das respostas, onde identificaram-se as práticas para cada unidade de conhecimento, observa-se quais as práticas mais adotadas, conforme Gráfico 10.

Gráfico 10 - Percentual de Adoção de Práticas de Capacitação



A prática de capacitação mais adotada foi *mentoring*, que consiste no consultor orientar e compartilhar com os profissionais da empresa-alvo da melhoria, suas experiências e conhecimentos nos processos de MPS. A segunda prática mais adotada foi

workshop, onde os consultores realizam um seminário de curta duração, apresentando técnicas e habilidades e demonstrando como estas podem ser aplicadas.

O uso de dinâmicas, onde um instrutor sugere uma atividade análoga a uma atividade técnica, e da prática de *coaching*, onde um profissional realiza um treinamento em técnicas específicas, ainda são poucos explorados por consultores. Destaca-se a necessidade de adoção destas práticas, devido aos benefícios inerentes à implementação destas.

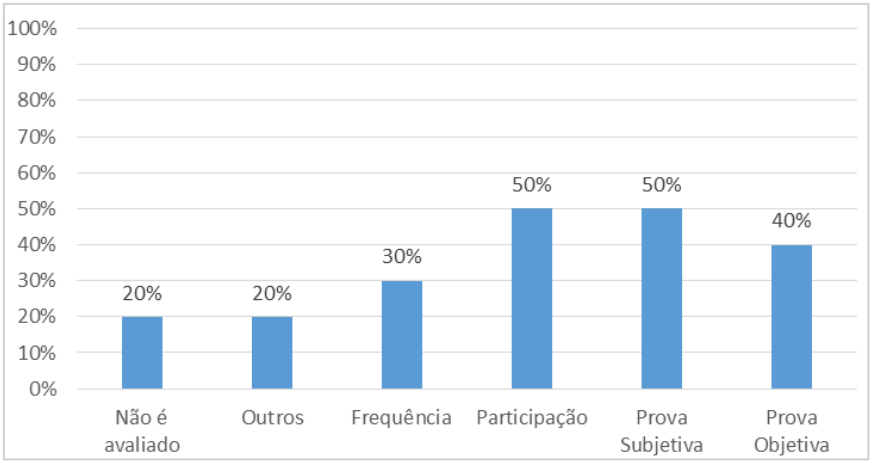
Os participantes de dinâmicas relatam que a interação entre os processos é melhor compreendida do que em exposições teóricas. Além disso, estas permitem aumentar o nível de interação ente os alunos e o facilitador, auxiliam na consolidação de conceitos a partir da vivência da teoria. Já a prática de *coaching* permite que um profissional especialista em determinados processos possa realizar um treinamento específico, diretamente com o aluno. Neste sentido, *coaching* talvez seja a prática mais efetiva do ponto de vista técnico.

Por fim, em relação às práticas de capacitação, destaca-se que 20% dos consultores adotam outras práticas de capacitação, como Cursos e Palestras. Estes cursos e palestras podem ser relacionados com a prática de *Workshop*. Apenas 10% dos processos não foram implementados pelos consultores.

4.2. Sobre as Estratégias de Avaliação

Em relação às estratégias de avaliação, observam-se os seguintes percentuais de adoção no Gráfico 11.

Gráfico 11 - Percentual de Adoção de Estratégias de Avaliação



As estratégias de avaliação mais adotadas foram provas subjetivas e a participação dos alunos. Acredita-se que essa adoção se deva ao fato de que estas estratégias permitem uma avaliação mais subjetiva das competências e habilidades dos alunos, sendo mais qualitativas do que as demais.

Observou-se um grande percentual de adoção de provas objetivas e frequência. Acredita-se que, para determinados processos, a realização de provas objetivas é a estratégia mais adequada, pois estes processos envolvem conceitos e fundamentos

essenciais que devem ser compreendidos pelos alunos. Além disso, no que diz respeito à frequência, assiduidade é um fator crítico em todo e qualquer tipo de capacitação.

Por fim, destaca-se que 20% dos consultores adotam outras estratégias de avaliação, como a realização de atividades práticas em um projeto. Estas atividades práticas podem ser associadas à estratégia de participação. Cerca de 20% dos treinamentos em processos não foram avaliados pelos consultores.

5. CONSIDERAÇÕES FINAIS

Em resumo, o objetivo deste levantamento foi "identificar as práticas de capacitação e estratégias de avaliação adotadas por consultores em MPS que atuam como professores de ES". Acredita-se que este objetivo foi atendido, pois diversas práticas foram identificadas e associadas com as unidades de conhecimento da ES.

A principal limitação deste levantamento consistiu na quantidade de participantes. Obteve-se uma baixa amostragem, considerando que os formulários de levantamento foram divulgados para 40 professores/consultores identificados no site da SOFTEX (*www.softex.br/*). No entanto, ressalta-se que, a fim de reduzir este viés da amostragem, realizou-se o levantamento nas regiões norte, nordeste e sudeste do Brasil.

No total, obtiveram-se respostas de 5 instituições implementadoras de MPS. Destaca-se ainda a média de experiência dos 10 consultores/professores entrevistados, sendo 13 anos e meio de docência e 11 anos atuando com consultoria em MPS.

A principal contribuição deste levantamento é fornecer à comunidade acadêmica da Engenharia de Software um conjunto de práticas que possam ser adaptados para o contexto acadêmico no ensino de ES. Espera-se que os alunos possam obter uma formação mais adequada, a partir da adoção de práticas de capacitação adotadas na indústria e que permitam desenvolvimento mais adequado de competências e habilidades em Engenharia de Software. Para os profissionais do mercado, a contribuição desta pesquisa será, a longo prazo, a inserção de profissionais mais bem preparados para atender as demandas do mercado.

Como trabalho futuro, a partir da identificação e da análise destas práticas de capacitação serão definidas estratégias de adequação destas para o ensino de ES. Pretende-se realizar esta customização através de um *framework* de ensino. Pretende-se, ainda, adotar a estratégia do modelo estagiado do CMMI-DEV, onde se definem quais áreas de processo serão foco da melhoria. Assim, alunos e professores poderão delimitar o escopo de seus esforços, reduzindo a quantidade de tópicos de ES ministrados para trabalhar o desenvolvimento de competências e habilidades em ES.

APÊNDICE D – AVALIAÇÃO DO PAINEL DE ESPECIALISTAS

1. INTRODUÇÃO

O modelo de ensino iterativo avaliado apresenta um conjunto de abordagens, técnicas, métodos e recursos que podem ser utilizados no processo de ensino-aprendizagem de Engenharia de Software (ES). Neste sentido, se caracteriza como um padrão ou plano que pode ser usado para guiar a especificação de currículos ou cursos da área de Computação, para selecionar materiais de instrução e para orientar as ações de um professor. De maneira mais específica, o modelo apresenta formas de ensino e aprendizagem que se destinam a apoiar o desenvolvimento das competências técnicas necessárias em alunos que pretendem atuar profissionalmente na área de ES.

A avaliação deste modelo ocorreu através de um painel de especialistas a fim de obter percepções de professores/pesquisadores que atuam na área. Os integrantes do painel se comprometeram em ler integralmente e analisar o modelo a fim de fornecer suas conclusões e recomendações. Durante este processo, os especialistas foram recomendados a adotar o mesmo rigor de outros métodos científicos de pesquisa, buscando (ao final) o consenso, mas não a ponto de eliminar todas as discordâncias.

Inicialmente, os especialistas foram convidados via e-mail, através do qual foi enviado o modelo de ensino e o formulário de avaliação, disponibilizado através de formulário eletrônico na plataforma *Google Docs*. Após a análise e avaliação individual do modelo, foi feita uma reunião de consenso entre os especialistas, onde os mesmos justificaram as notas atribuídas na avaliação. Esta reunião buscou resolver as discordâncias, possibilitando o desenvolvimento de um consenso final do painel de especialistas em relação à documentação e ao uso do modelo de ensino proposto. Como resultado, foi emitido um parecer conclusivo da avaliação, apresentado a seguir.

2. DOCUMENTAÇÃO DO MODELO

2.1 Adequação

O texto da documentação está adequado para o tema abordado pelo modelo (Ensino de Engenharia de Software)?

2 SIM

Avaliador 1: Sem considerações.

Avaliador 2: O texto está muito bem escrito e referenciado.

1 NÃO

Avaliador 3: Poderia ser mais objetivo e mais claro. Há muita repetição e muito texto, o que dificulta quem for tentar utilizá-lo.

2.2 Clareza

O texto da documentação está claro e objetivo?

1 SIM

Avaliador 2: Sem considerações.

2 NÃO

Avaliador 1: Não encontrei no documento uma seção que contenha "o modelo" (aparentemente este conteúdo é encontrado na Seção 3). Nesta documentação, eu esperaria que "o modelo" fosse a primeira coisa a ser apresentada, e as justificativas fossem apresentadas posteriormente.

Avaliador 3: Poderia ser bem mais claro.

2.3 Completude

O texto da documentação apresenta os componentes necessários à especificação de um modelo de ensino?

3 SIM

Avaliador 1: Sem considerações.

Avaliador 2: Sem considerações.

Avaliador 3: Sem considerações.

2.4 Consistência

Os componentes estão consistentes com o foco do modelo de ensino?

3 SIM

Avaliador 1: Sem considerações.

Avaliador 2: Sem considerações.

Avaliador 3: Sem considerações.

2.5 Corretude

O texto da documentação está claro e objetivo?

3 SIM

Avaliador 1: Com o nível de superficialidade que olhei o documento para responder este formulário, não acho que a minha resposta seja suficientemente confiável.

Avaliador 2: Sem considerações.

Avaliador 3: Sem considerações.

2.6 Detalhamento

O nível de detalhes é suficiente para fornecer uma base adequada para o uso do modelo de ensino?

1 SIM

Avaliador 2: Achei o capítulo 3 resumido, porém, considerando que tenho que ler a Apêndice A para entender o modelo considero suficiente.

2 NÃO

Avaliador 1: Seria interessante que este documento contivesse mais recomendações de uso, e mais exemplos. Talvez um *survey* com professores que fazem coisas aderentes ao modelo possa coletar uma quantidade suficiente de recomendações/exemplos para enriquecer mais o documento.

Avaliador 3: Poderia se utilizar de mais representações gráficas e tabelas.

3. USABILIDADE DO MODELO

3.1 Facilidade de Uso

O modelo é fácil de usar?

1 Concorda

Avaliador 2: A lógica é fácil de compreender, mas para colocar em prática, cada uma das etapas exige um esforço aparentemente grande do professor na preparação do seu material, bem como um conjunto de conhecimentos e comportamentos não-triviais

(como *coaching*, por exemplo). Não tenho certeza se apenas a disposição de um professor para usar o modelo é suficiente para que ele seja executado apropriadamente.

2 Discordam

Avaliador 1: Considero que é fácil, porém depende da disposição do professor.

Avaliador 3: Sem considerações.

3.2 Flexibilidade do Modelo

O modelo permite flexibilidade no seu uso?

2 Concordam Totalmente

Avaliador 1: Sem considerações.

Avaliador 2: Sem considerações.

1 Concorda Parcialmente

Avaliador 3: Sem considerações.

3.3 Adequação do Modelo

O modelo é adequado para ser usado pela maioria dos professores?

2 Concordam Parcialmente

Avaliador 1: Acho que cabe uma reflexão sobre a adequação do modelo para disciplinas mais (ou menos) técnicas.

Avaliador 2: Sem considerações.

1 Discorda

Avaliador 3: Há vários tipos de professores. Poderia haver orientações de acordo com a categoria em que o professor se enquadra.

3.4 Esforço Envolvido

Qual o grau de esforço envolvido para um professor usar o modelo?

1 - Esforço Moderado

Avaliador 2: Sem considerações.

2 - Muito Esforço

Avaliador 1: O esforço aqui consiste na preparação e atualização de um material que faça sentido continuamente. Disciplinas puramente expositivas, embora possam ser consideradas ineficazes e chatas, são absolutamente convenientes para o professor, e exigem esforço de preparação apenas 1 vez (e um esforço insignificante para atualização dela). Disciplinas que adotam o seu modelo ainda podem inserir uma série de incertezas que podem gerar um risco grande de desconforto do professor.

Avaliador 3: Sem considerações.

3.5 Habilidades Requeridas

Qual o grau de habilidades requeridas para um professor usar o modelo?

1 - Habilidades Moderadas

Avaliador 2: Sem considerações.

2 - Habilidades Avançadas

Avaliador 1: Sem considerações.

Avaliador 3: Sem considerações.

3.6 Restrições de Uso

Qual o grau de restrições para usar o modelo?

2 - Restrições Moderadas

Avaliador 2: Considerando o apoio do profissional da indústria ou egresso e o planejamento das dinâmicas, das leituras e das discussões e projetos práticos.

Avaliador 3: Sem considerações.

1 - Não consigo avaliar

Avaliador 1: A pergunta não está muito clara.

4. PARECER CONCLUSIVO

4.1 Utilidade do Modelo

O quão útil você considera o modelo apresentado no apoio ao ensino de Engenharia de Software?

1 - Moderadamente Útil

Avaliador 1: O modelo é suficientemente sintético, e representativo do conjunto de teorias mencionadas na fundamentação. Se um professor não adere aos fundamentos do modelo, ele não é útil *at all*.

2 - Muito Útil

Avaliador 2: Sem considerações.

Avaliador 3: Poderia guiar melhor o professor na sua utilização, visto que professores de engenharia de software não têm formação em pedagogia.

4.2 Recomendação do Modelo

O quanto você recomendaria esse modelo para uso em sala de aula no ensino de Engenharia de Software?

1 - Fortemente

Avaliador 1: Sem considerações.

2 - Largamente

Avaliador 2: Sem considerações.

Avaliador 3: Desde que sejam melhorados a sua facilidade de uso e o seu conteúdo de apoio.

4.3 Pontos Fracos do Modelo

De maneira geral, quais são os pontos fracos do modelo avaliado?

Avaliador 1: Mencionei alguns nas respostas anteriores;

Avaliador 2: Não achei nenhum ponto fraco que deveria ser mencionado;

Avaliador 3: Baixa usabilidade, falta de clareza e objetividade.

4.4 Pontos Fortes do Modelo

De maneira geral, quais são os pontos fortes do modelo avaliado?

Avaliador 1: Ser um material que o professor, que não é especialista em pedagogia, pode contar com ele.

Avaliador 2: O aluno como centro do estudo / - a utilização de profissionais da indústria para motivar o ensino da matéria / - o ciclo PDCA sendo utilizado para verificar

a eficácia do ensino (retrospectiva)/ - responsabilidade de aprendizado também para o aluno.

Avaliador 3: Foram citados anteriormente.

4.5 Considerações Finais

Quais são as recomendações gerais do avaliador(a) para os proponentes do modelo?

Avaliador 1: Mencionei alguns nas respostas anteriores. Não me lembro de todos.

Avaliador 2: Achei que o modelo representa bem o que eu penso sobre o ensino nos dias atuais. O problema que pode vir acontecer é a mudança de paradigma para aqueles professores que estão acostumados com o ensino tradicional.

Avaliador 3: Melhorar bastante a facilidade de uso do modelo e o conteúdo, pois muitos professores não terão paciência de ler e reler o modelo até poder utilizá-lo.

APÊNDICE E – TCLE DO EXPERIMENTO CONTROLADO

1. Você está sendo convidado(a) a participar de um estudo de caso relativo a pesquisa de doutorado intitulada "***Um Modelo Iterativo para o Ensino de Engenharia de Software Baseado em Abordagens Focadas no Aluno e Práticas de Capacitação da Indústria***".
2. Esta pesquisa tem como intuito observar a aplicação de um modelo de ensino para coletar dados sobre o ensino-aprendizagem de Engenharia de Software.
    a. Você foi selecionado para participar de um experimento, no qual assumirá responsabilidades inerentes a um discente e desempenhará atividades relacionadas ao modelo de ensino que será adotado na disciplina.
    b. Sua participação nesta pesquisa consistirá em interagir com os demais estudantes e professores. Após as realizações de atividades técnicas e interações pessoais, você poderá relatar suas experiências, pontos positivos, pontos negativos e oportunidades de melhoria para o modelo adotado. Este relato será por meio de questionários e entrevistas.
3. A sua participação neste experimento controlado envolve riscos mínimos (como desconforto moral, ético ou físico) e é voluntária (você não terá nenhum benefício financeiro por participar desse estudo), podendo recursa-se a participar ou retirar seu consentimento, em qualquer fase da pesquisa, sem penalização alguma.
    a. A qualquer momento você pode desistir de participar e retirar seu consentimento.
    b. Sua recusa não trará nenhum prejuízo em sua relação com o pesquisador ou com a Instituição.
4. A pesquisa ocorrerá no Laboratório de Sistemas de Informação que fica nas dependências do Campus Universitário do Tocantins/Cametá – Universidade Federal do Pará (UFPA), em um horário previamente definido. Durante a pesquisa, você será assistido pelos condutores do experimento. Os recursos a serem utilizados são diversos, como computadores, *data-show*, quadro branco. O tempo de cada seção do estudo de caso não ultrapassará 2 horas.
5. Antes e durante o curso da pesquisa, você poderá solicitar esclarecimentos a respeito dos procedimentos ou qualquer outra questão relacionada a esta pesquisa.
6. Seus dados pessoais envolvidos na pesquisa serão confidenciais.
    a. Os dados coletados no experimento controlado serão analisados e todos os participantes receberão por e-mail os resultados da Análise dos Dados coletados,

por meio de um artigo publicado ou um capítulo específico da Tese de Doutorado sobre o experimento realizado.

b. Toda e qualquer informação coletada durante o estudo é tratada como confidencial. Os dados não serão divulgados de forma a possibilitar sua identificação. Para isso, usaremos nomes e siglas fictícias para a apresentação de dados.

7. Os resultados obtidos através desta pesquisa serão utilizados para propor melhorias no ensino de Engenharia de Software que poderão beneficiar professores e estudantes de cursos de graduação em Computação de instituições públicas e privadas do Brasil.
8. Você receberá uma cópia deste termo onde consta o contato dos pesquisadores, podendo tirar suas dúvidas sobre o projeto e sua participação, agora ou a qualquer momento. Você poderá entrar em contato com os pesquisadores no endereço Trav. Padre Antônio Franco, 2617 – Matinha – Cametá-PA, ou pelo telefone (91) 980656772.

Responsáveis pela Pesquisa	**Email**
Carlos dos Santos Portela	csp3@cin.ufpe.br
Alexandre Marcos Lins de Vasconcelos	amlv@cin.ufpe.br
Sandro Ronaldo Bezerra Oliveira	srbo@ufpa.br

Cametá, 27 de Junho de 2017.

Carlos dos Santos Portela

Assinatura do Pesquisador Responsável

ASSINATURA DO TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO (TCLE)

FASE 1 DO EXPERIMENTO – GERENCIAMENTO DE PROJETOS

Nome do Aluno	Turma
Carlos Alexandre Pinheiro de Souza	BSI 2014
Joiner dos Santos Sá	BSI 2014
Jefson Farias de Farias	BSI 2014
Carlos Augusto José J. Machado	BSI 2014
Márcio Heniz P. Furtado Junior	BSI 2014
Ênio de Jesus Pontes Monteiro	BSI 2014
Francisco Alencar Vasconcelos	BSI 2014
Marcos Antônio Nascimento Figueiredo Júnior	BSI 2014
Jessé Alves Gonçalves	BSI 2014
Gracilene Américo Viana	BSI 2014
Eliezer Miranda Coelho	BSI 2014
Raiza Portilho Nunes	BSI 2014
Shirley Evangelista Franca	BSI 2014
Antonio Carlos Lima Sousa	BSI 2014
Edilson Carlos S. do Nascimento	BSI 2014
Valdir Nunes Sacramento	BSI 2014

ASSINATURA DO TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO (TCLE)

FASE 2 DO EXPERIMENTO – ENGENHARIA DE REQUISITOS

Nome do Aluno	Turma
Valdir Nunes sacramento	2014
Luis Paulo Moreira de Souza	SI 2015
Alfredilson de Freitas Alves.	SI 2015
Kelvin cristian Rodrigues siqueira	SI 2015
Luis Anderson R. Freitas	SI 2015
Jociel Garcia Borges	SI 2015
Marcos Vinicius Dos Passos Oliveira	SI 2015
Benedito dos Prazeres Gonçalves	SI 2015
Eduardo de Almeida Rodrigues	SI 2015
Alef Pontes dos Santos	SI 2015
Lourenzo Lopes Cardoso	SI 2015
Márcio Lima Modesto	SI 2014
Mario Antônio N. Figueiredo Júnior	SI 2015
João Mauricio R. Santos Marques	SI 2015
Manoel do Carmo S. Júnior	SI 2012
Marco Antonio Silva Furtado	SI 2012
Ysamar dos Santos Valente	SI 2015
Elton Donato Lopes dos Santos.	SI 2015.
Rodrigo da Silva Assunção	SI 2015
José A. S. Nogueira	SI 2015
Maxwell Medeiros de Souza	SI 2015

APÊNDICE F – RELATÓRIO DO EXPERIMENTO CONTROLADO

1. DEFINIÇÃO DO PROBLEMA

A disciplina de Engenharia de Software (ES) é tratada no ensino formal no nível de graduação e pós-graduação ou por meio de treinamentos profissionais de curta duração [1]. De maneira geral, os cursos da área de Computação objetivam desenvolver nos estudantes a capacidade para aplicar seus conhecimentos de forma independente e desenvolver competências e habilidades específicas, como trabalho em grupo, comunicação escrita e verbal e resolução de problemas [2][3]. Para tal, devem utilizar uma abordagem para o ensino de ES que contemple o conteúdo a ser ministrado, as habilidades a serem desenvolvidas, as atividades práticas e laboratoriais e as metodologias de ensino-aprendizagem [2].

Neste sentido, os currículos de referência da área (ACM/IEEE e SBC) ressaltam que essas competências emergem através do estudo teórico destas unidades e da aplicação prática de seus conceitos [2][3]. Para atender este propósito, recomenda-se a configuração de um ambiente prático de projeto de software [3]. Assim, o estudante poderá aplicar seu conhecimento teórico, pois um projeto prático irá requerer que este resolva problemas técnicos, trabalhe em equipe e desenvolva habilidades de comunicação interpessoal [2].

No entanto, nem todos os cursos oferecem essa oportunidade aos alunos de realizarem atividades práticas [4]. Segundo Prikladnicki et al. [5], muitos professores adotam abordagens tradicionais para ensinar ES, como aulas expositivas, que acabam sendo pouco eficientes, pois são centradas apenas no professor.

Existe um consenso entre os pesquisadores da área de que abordagens práticas são as mais indicadas para o ensino de Engenharia de Software [1][4][5]. Adicionalmente, estes autores destacam que esse ensino deve ser mais centrado no aluno, a fim de aumentar sua motivação, participação e, consequentemente, sua aprendizagem. Uma preocupação atual é entender como usar essas abordagens eficazmente para contribuir com a formação e aumento da competência dos engenheiros de software [1].

Dado o presente cenário, onde a maioria dos professores segue uma abordagem tradicional e que alguns professores adotam abordagens práticas, onde o aluno é o foco do processo de ensino-aprendizagem, se faz necessário comparar estas duas abordagens.

Desta forma, apresenta-se na Tabela 1, o objetivo de pesquisa, conforme *template* apresentado por Wohlin et al. [6] e baseado na estrutura *Goal-Question-Metric* (GQM):

Tabela 1 – Objetivo de Pesquisa

Analisar	o uso de uma abordagens prática de ensino
Com o propósito de	avaliar a sua efetividade no desenvolvimento de competências técnicas em Engenharia de Software
Com relação ao	uso de uma abordagem tradicional de ensino
Do ponto de vista de	um professor da área de Engenharia de Software
No contexto de	uma instituição de ensino superior (acadêmico).

A fim de contextualizar a execução do experimento controlado que busca atingir este objetivo, a Seção 2 deste relatório apresenta a fundamentação teórica desta pesquisa, citando os principais trabalhos relacionados e suas contribuições para a realização deste experimento.

2. FUNDAMENTAÇÃO TEÓRICA

Nesta pesquisa, o termo abordagem de ensino baseia-se no mapeamento sistemático realizado por Santos et al. [1] que conceitua “propostas para o ensino de Engenharia de Software”. Assim, abrange planejamento pedagógico, o conteúdo a ser ministrado, aspectos metodológicos, como métodos de ensino, além de aspectos tecnológicos que viabilizem a aplicação destes.

De acordo com Prikladnicki et al. [5], as abordagens mais comuns para ensinar ES incluem aulas expositivas, aulas de laboratório, entre outras. Entretanto, algumas abordagens alternativas vêm sendo adotadas por professores de algumas instituições, que promovem maior participação dos estudantes nas aulas de ES [1][7], com estratégias que trocam aulas expositivas por discussões de casos práticos [8], dinâmicas de grupo [9], jogos e simuladores educacionais [10][11][12] e projetos práticos, onde os estudantes devem desenvolver um sistema, muitas vezes envolvendo um cliente real [3][13].

Dentre estas abordagens, destacam-se as que baseiam o ensino de ES em discussões de casos práticos e/ou na realização de projetos práticos. Quanto aos métodos de ensino, observa-se que os métodos mais focados no aluno, e que promovem uma participação mais ativa nas aulas, têm potencial para aumentar o interesse dos estudantes,

motivá-los e melhorar a aprendizagem no nível de aplicação dos conceitos [5]. Estas abordagens também são denominadas humanistas [14].

De acordo com Coutinho e Bezerra [9], estudos de casos práticos normalmente são realizados de maneira a enfatizar a aprendizagem de aspectos tecnológicos, tais como linguagens de especificação, métodos e ferramentas de apoio. Esta abordagem permite fixar diversos conceitos de ES através de atividades práticas e da realização de dinâmicas. Geralmente, estas práticas envolvem aspectos comportamentais, ajudando a trabalhar os diversos perfis envolvidos na ES, semelhante às situações encontradas em ambientes reais de desenvolvimento de software na indústria.

As recomendações da ACM/IEEE [3] sugerem que sejam realizados nas disciplinas de ES atividades práticas do início até o fim, onde grupos de alunos planejam e executam um projeto de software durante o semestre inteiro. Este projeto é denominado *capstone project* e deve requerer que os estudantes trabalhem em equipe para desenvolver um software utilizando, tanto quanto possível, um ciclo de vida. Além disto, estes projetos devem ser suficientemente desafiadores para requerer que os estudantes usem técnicas de engenharia efetivamente e desenvolvam e pratiquem suas habilidades de comunicação e resolução de problemas.

No entanto, a ACM/IEEE [3] destaca, através das suas pesquisas sobre o ensino de ES, que há evidências crescentes de que os estudantes aprendem a aplicar os princípios de ES de forma mais eficaz através de abordagens iterativas, onde terão a oportunidade de realizar suas atividades em um ciclo de desenvolvimento, avaliar o seu trabalho e, em seguida, aplicar o conhecimento adquirido em um próximo ciclo de desenvolvimento. Modelos de ciclo de vida ágeis inerentemente providenciam essas oportunidades.

Neste contexto, Gary et al. [13] sugerem que a trajetória de aprendizagem deva ser espiral, com componentes revisitados e com níveis de detalhamento e interconexões. Este modelo pedagógico busca acelerar o desenvolvimento de competências nos estudantes através do entendimento e compreensão para aplicar conhecimentos de ES. Desta forma, possui um currículo flexível e centrado em projeto. O objetivo é integrar experiências contextualizadas de projetos com conceitos fundamentais de ES através de uma abordagem denominada *Software Enterprise*.

Esta proposta foi maturada durante 9 anos de aplicação na *Arizona State University Polytechnic*, onde foram desenvolvidos os produtos de trabalho para apoiar a

metodologia centrada em projeto [13]. Na abordagem *Software Enterprise* emprega-se um modelo de aprendizagem que combina aula tradicional, atividades em grupo, aprendizagem centrada em problema (PBL), reflexão e prática.

Nesta abordagem, inicialmente os alunos realizam leituras técnicas em relação a uma área da ES. Em seguida, os alunos assistem a diversas palestras para criar consciência e compreensão e colocam os conceitos de aula em prática através de sessões de laboratório em grupo a cada semana. Por fim, os alunos integram as habilidades adquiridas durante o processo para o projeto de software da disciplina de ES, refletem sobre essa rápida experiência de aprendizagem, e se envolvem em grupos de discussão sobre as lições aprendidas. Após a conclusão deste ciclo iterativo, inicia-se um novo ciclo para uma nova unidade de conhecimento de ES.

3. PLANEJAMENTO DO EXPERIMENTO

De acordo com Easterbrook et al. [15], uma pré-condição para conduzir um experimento controlado é ter claro quais hipóteses serão observadas. Estas hipóteses irão guiar todas as etapas do experimento, incluindo quais variáveis serão mensuradas. Desta forma, o experimento conduzido nesta pesquisa buscou observar as seguintes hipóteses:

H_0: *A abordagem humanista de ensino permite o desenvolvimento de competências técnicas em Engenharia de Software em nível de aplicação de maneira menos efetiva, ou igual, do que a abordagem tradicional de ensino.*

Na hipótese H_0, o tratamento é a "abordagem humanista de ensino" e o controle é a "abordagem tradicional de ensino". Neste cenário, a hipótese alternativa a ser observada seria:

H_1: *A abordagem humanista de ensino permite o desenvolvimento de competências técnicas em Engenharia de Software em nível de aplicação de maneira mais efetiva do que a abordagem tradicional de ensino.*

Essas hipóteses podem ser formalizadas a partir da seguinte representação:

H_0 = abordagem humanista de ensino ≤ abordagem tradicional de ensino
H_1 = abordagem humanista de ensino > abordagem tradicional de ensino

Na análise dessas duas hipóteses, foi observada a variável dependente competência técnica em Engenharia de Software, que foi medida usando como referência

o processo de avaliação do SWECOM [16], para determinar se a variável independente abordagem de ensino possui efeito no resultado final. Da mesma forma, as variáveis independentes abordagem humanista de ensino e abordagem tradicional de ensino foram variáveis manipuladas no experimento a fim de avaliar a sua influência na variável dependente.

A. Objetivos

A fim de realizar a observação destas teorias, os objetivos deste experimento são:

Objetivo 1: *Analisar* o uso de uma abordagem humanista de ensino focada no aluno

Com o propósito de avaliar a sua efetividade no desenvolvimento de competências técnicas em Engenharia de Software

Com relação ao uso de uma abordagem tradicional de ensino focada no professor.

Objetivo 2: *Analisar* o uso de um modelo de ensino baseado em abordagens de ensino focadas no aluno e práticas de capacitação da indústria

Com o propósito de avaliar a sua efetividade no desenvolvimento de competências técnicas em Engenharia de Software

Com relação ao uso de uma abordagem tradicional de ensino centrada em conteúdo.

B. Participantes

Como público-alvo do experimento controlado, definiu-se:

I Professor(a) que está lecionando a disciplina da área de ES;

II Aluno(a)s que estão cursando a disciplina da área de ES.

Este experimento controlado foi aplicado em um curso de graduação em Sistemas de Informação da Universidade Federal do Pará (UFPA), Campus de Cametá, durante o primeiro semestre de 2017. As disciplinas-alvo foram “Gerência de Projetos de Software”, cuja unidade de conhecimento foco do experimento foi Gerenciamento de Projetos, onde participaram 12 alunos. Já na disciplina de “Análise e Projeto de Sistemas I”, cuja unidade foco foi Engenharia de Requisitos, participaram 18 alunos.

Foi apresentado aos alunos destas disciplina um Termo de Consentimento Livre e Esclarecido (TCLE) a fim de obter o comprometimento com a pesquisa. Este termo explica o propósito e as etapas do experimento controlado. Somente os alunos que assinaram este termo participaram do experimento. É importante destacar que estes

estudantes já tinham uma experiência acadêmica prévia, pois já haviam cursado disciplinas da área de Engenharia de Software.

C. Instrumentos

Foram utilizados como instrumentos para aplicação do experimento entrevistas estruturadas, questionários auto administrados, dicionários de dados e mapas conceituais. Estes instrumentos foram aplicados tanto no grupo de controle, que seguiu uma abordagem tradicional, quanto nos grupos do experimento, que seguiram abordagens focadas no aluno e práticas de capacitação da indústria. A abordagem tradicional foi representada pela atual abordagem adotada pelo professor na disciplina de ES, e as abordagens focadas no aluno e práticas de capacitação da indústria foram seguidas através de um modelo de ensino iterativo proposto pelos pesquisadores.

As entrevistas estruturadas foram baseadas no formulário de avaliação de competências técnicas em ES do modelo SWECOM [16], aplicados no início e no final da disciplina, a fim de medir o desenvolvimento destas competências. Já os questionários utilizaram as mesmas questões de pesquisa do *survey* realizado em Portela, Vasconcelos e Oliveira [7] relacionadas à aprendizagem de tópicos. Estes questionários foram disponibilizados no *Google Forms* (*https://www.google.com/forms*), também aplicados antes do início e após a conclusão da disciplina.

Por fim, os dicionários de dados e mapas conceituais foram utilizados como instrumentos de avaliação qualitativa da aprendizagem dos alunos. Os dicionários de dados criados pelos alunos permitem analisar os conceitos absorvidos por estes em relação aos tópicos de ES. Um dicionário de dados consiste em um repositório central de informações acerca de conceitos de uma área específica. Sendo assim, são definidos como representações textuais, que descrevem palavras e conceitos.

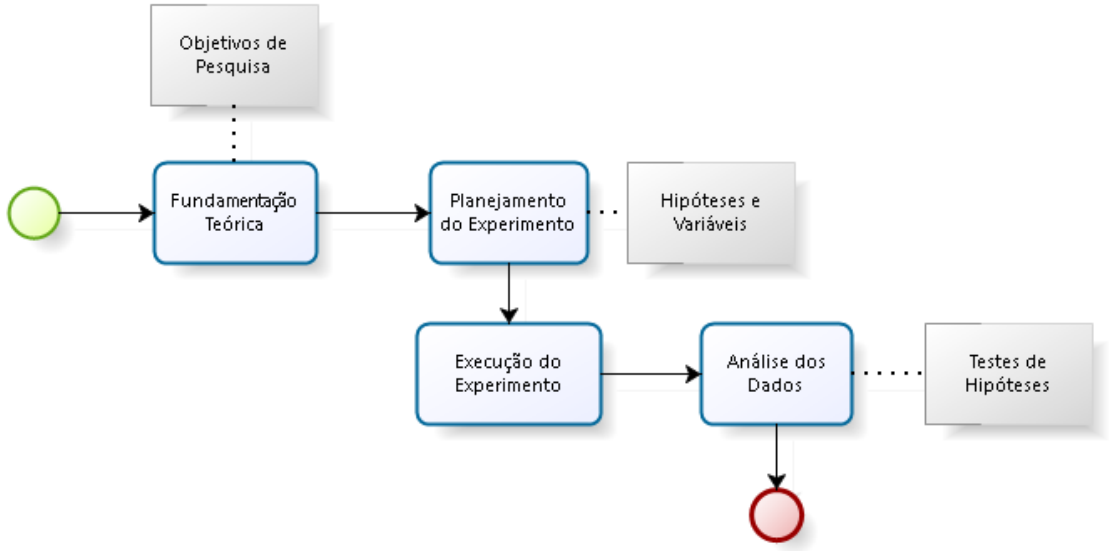
Já os mapas conceituais permitem analisar como os alunos relacionam estes conceitos entre si. Mapas conceituais são definidos como estruturas esquemáticas que representam conjuntos de conceitos dispostos em uma espécie de rede de proposições. Desta forma, permitem apresentar de forma mais clara a exposição do conhecimento, organizando-o segundo a compreensão cognitiva do seu idealizador. Esta apresentação é feita a partir de representações gráficas, que indicam relações entre palavras e conceitos.

D. Design do Experimento

O protocolo de pesquisa definido para este experimento seguiu os *guidelines* de Jedlitschka, Ciolkowski e Pfahl [17]. Quanto ao *design* de coleta de dados, este experimento se classifica como 3x3, onde três grupos (amostras da população de estudo) receberam três tratamentos distintos e a coleta dos dados ocorre em dois momentos distintos no tempo (início e fim da disciplina). Neste tipo de experimento, os participantes são selecionados de maneira aleatória, onde define-se um grupo de controle e dois grupos experimentais [6].

A fim de planejar e executar o experimento controlado desta pesquisa, seguiram-se 4 (quatro) etapas, conforme ilustra a Figura 1.

Figura 1 – Etapas da Realização do Experimento Controlado



Inicialmente, na *Etapa I – Fundamentação Teórica* foram identificados os problemas, objetivos e contexto de pesquisa a fim de identificar as informações necessárias para as próximas etapas do experimento controlado. Em seguida, foram identificados os estudos relacionados às hipóteses e variáveis envolvidas a fim de fundamentar o experimento. Nesta etapa, identificaram-se as abordagens de ensino de Gnatz et al. [18] e Garg e Varma [19], que propõem a adaptação de práticas da indústria para o contexto acadêmico, a análise de abordagens focadas no aluno feita por Prikladnicki et al. [5], além do modelo pedagógico de Gary et al. [13]. Na *Etapa II – Planejamento do Experimento* definiram-se os objetivos do experimento, a população de estudo, a operacionalização das variáveis, o *design* e os procedimentos de análise de dados, neste caso, a análise de variância ANOVA [15].

Na *Etapa III – Execução do Experimento*, realizou-se o treinamento dos professores no modelo (abordagem humanista de ensino) em Abril de 2017. Em seguida, foram aplicados questionários pré e pós-aplicação das unidades de conhecimento de ES a fim de analisar a aprendizagem dos alunos. Estes questionários foram aplicados a fim de identificar o conhecimento prévio dos alunos e formar os grupos do experimento de forma nivelada. Por fim, na *Etapa IV – Análise dos Dados* buscou-se identificar o grau de aprendizagem dos alunos nas duas abordagens de ensino (tradicional e humanista), de acordo com as respostas fornecidas nos questionários e entrevistas dos professores e alunos participantes do experimento.

4. EXECUÇÃO DO EXPERIMENTO

Inicialmente, os professores da disciplina receberam o treinamento referente ao uso do modelo iterativo de ensino. Em seguida, na primeira aula da disciplina, o planejamento do experimento foi apresentado aos alunos que tiraram dúvidas em relação ao mesmo. Após a apresentação da abordagem de ensino, estes assinaram o TCLE concordando em participar do experimento controlado.

O experimento na disciplina de "Gerência de Projetos de Software" foi realizado durante o período de 26 de Abril a 13 de Maio de 2017 no Laboratório de Informática que fica nas dependências da UFPA Cametá, no período das 14h00 às 18h00. A turma foi dividida em 3 (três) grupos. Esses grupos foram divididos de acordo com os resultados obtidos no questionário de percepção de conhecimento dos tópicos da unidade de Gerenciamento de Projetos. Cada grupo foi composto por 4 (quatro) alunos.

O Grupo A seguiu a abordagem tradicional, sendo o grupo de Controle, o Grupo B seguiu a abordagem humanista através das práticas focadas no aluno do modelo de ensino, sendo o grupo do Tratamento 1. O Grupo C também seguiu a abordagem humanista, mas além das práticas focadas no aluno adotou as práticas de capacitação da indústria, sendo o grupo do Tratamento 2. Desta forma, durante a pesquisa, os alunos do Grupo C foram auxiliados por alunos veteranos que realizaram *mentoring* e pelo professor que realizou *coaching*.

Ao final da disciplina, os alunos preencheram novamente o questionário sobre o grau de aprendizado obtido para os tópicos de ES, realizaram a entrevista com especialista a fim de identificar as competências obtidas, e criaram novos dicionários de dados e

mapas conceituais. O objetivo foi comparar as respostas fornecidas pré e pós-disciplina, ou seja, a aprendizagem adquirida a partir do seguimento do controle e tratamentos.

Foi realizado uma segunda fase do experimento, que seguiu o mesmo protocolo e passos que o anteriormente relatado. Esse experimento envolveu a disciplina de "Análise e Projeto de Sistemas I" durante o período de 27 de Junho a 14 de Julho de 2017 no mesmo Laboratório de Informática da UFPA Cametá. Essa segunda fase do experimento foi realizado buscando reforçar os resultados obtidos no primeiro e aumentar a população do estudo. Assim, participaram da avaliação do modelo 2 professores e 30 alunos.

5. ANÁLISE DE DADOS

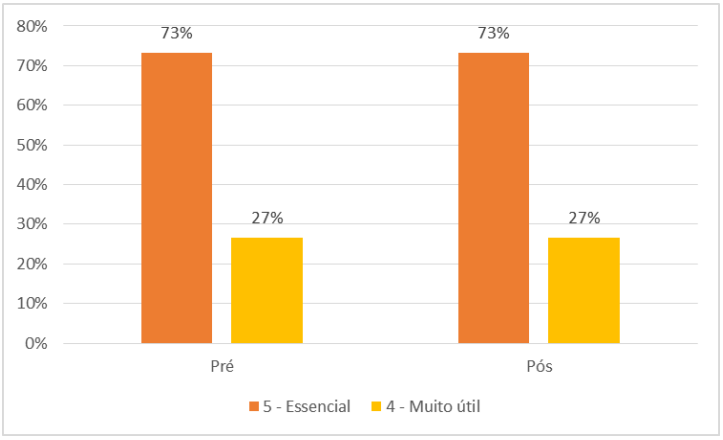
Esta seção apresenta a análise da média dos dados das 2 fases do experimento realizado, a fim de sintetizar os resultados e as conclusões.

5.1. Dados Estatísticos

A) Utilidade da área de Engenharia de Software

Em relação a percepção inicial (pré-experimento) do Grupo de Controle, que seguiram a abordagem de ensino tradicional, quanto a utilidade da área de Engenharia de Software para sua formação profissional, 73% considera a disciplina essencial, enquanto 27% muito útil, conforme Gráfico 1.

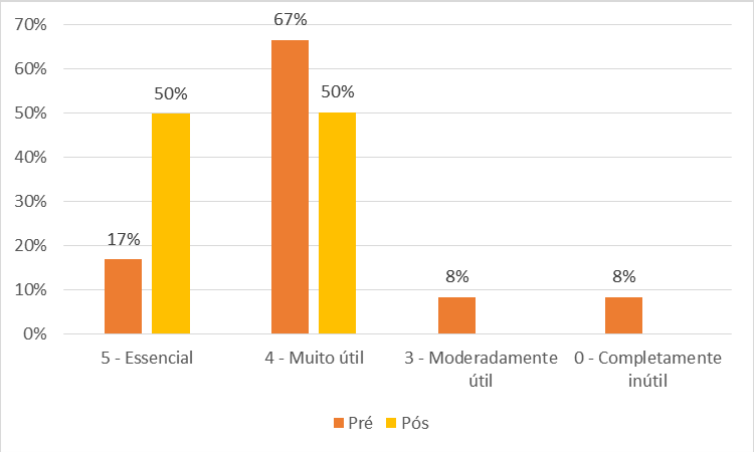
Gráfico 1 - Percepção do Grupo Controle sobre a Utilidade da Engenharia de Software



Após a conclusão do experimento, ou seja, após o uso da abordagem tradicional, 73% deste mesmo grupo de alunos responderam que consideram a área de ES essencial e 27% muito útil. Portanto, observar-se que não houve mudança de percepção de utilidade nesta abordagem.

Já o Grupo do Tratamento 1 (abordagem humanista - práticas focadas no aluno) responderam, no pré-experimento, que 17% considerava a área de ES essencial, 67% muito útil, 8% moderadamente útil e 8% completamente inútil, conforme Gráfico 2.

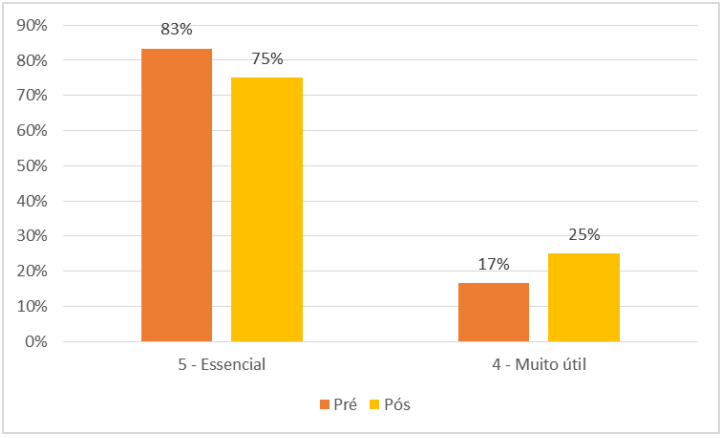
Gráfico 2 - Percepção do Grupo Tratamento 1 sobre a Utilidade da Engenharia de Software



Após a conclusão do experimento, ou seja, após o uso das práticas focadas no aluno, 50% deste mesmo grupo de alunos responderam que consideram a área de ES essencial e 50% muito útil. Portanto, observar-se que houve um aumento considerável na percepção de utilidade neste grupo.

Por fim, o Grupo do Tratamento 2 (abordagem humanista - práticas focadas no aluno e práticas de capacitação da indústria) responderam, no pré-experimento, que 83% considerava a área de ES essencial e 17% muito útil, conforme Gráfico 3.

Gráfico 3 - Percepção do Grupo Tratamento 2 sobre a Utilidade da Engenharia de Software



Após a conclusão do experimento, ou seja, após o uso das práticas focadas no aluno e práticas de capacitação da indústria, 75% dos alunos deste grupo responderam que consideram a área essencial e 25% muito útil. Observar-se que houve uma pequena redução no grau “Essencial” e um ligeiro aumento no grau “Muito útil”.

B) Aprendizagem dos Tópicos de Engenharia de Software

A aprendizagem dos tópicos de Engenharia de Software foi operacionalizada através da escala *Likert* de 0 até 5 para caracterizar os seguintes graus de aprendizagem:

0 - Não sei absolutamente nada

1 - Sei vagamente

2 - Sei o básico

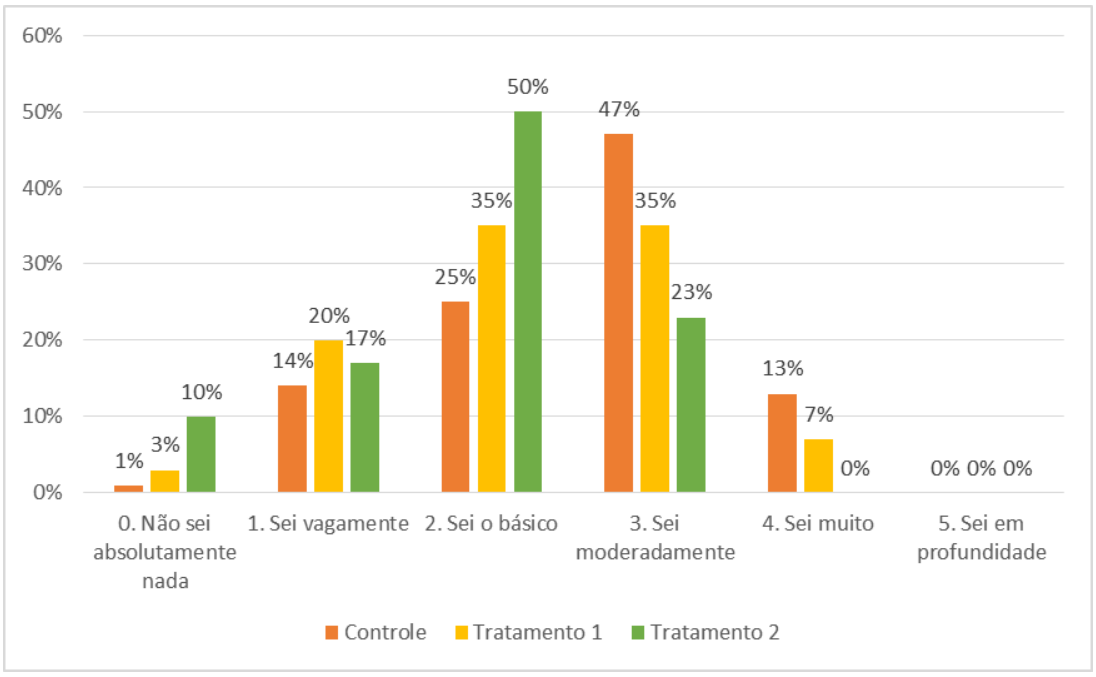
3 - Sei moderadamente

4 - Sei muito

5 - Sei em profundidade (sou especialista)

Assim, a partir da análise das respostas obtidas no experimento, o Gráfico 4 apresenta o percentual de respostas obtidas para cada grau de conhecimento, segundo os 3 grupos antes de iniciar o experimento.

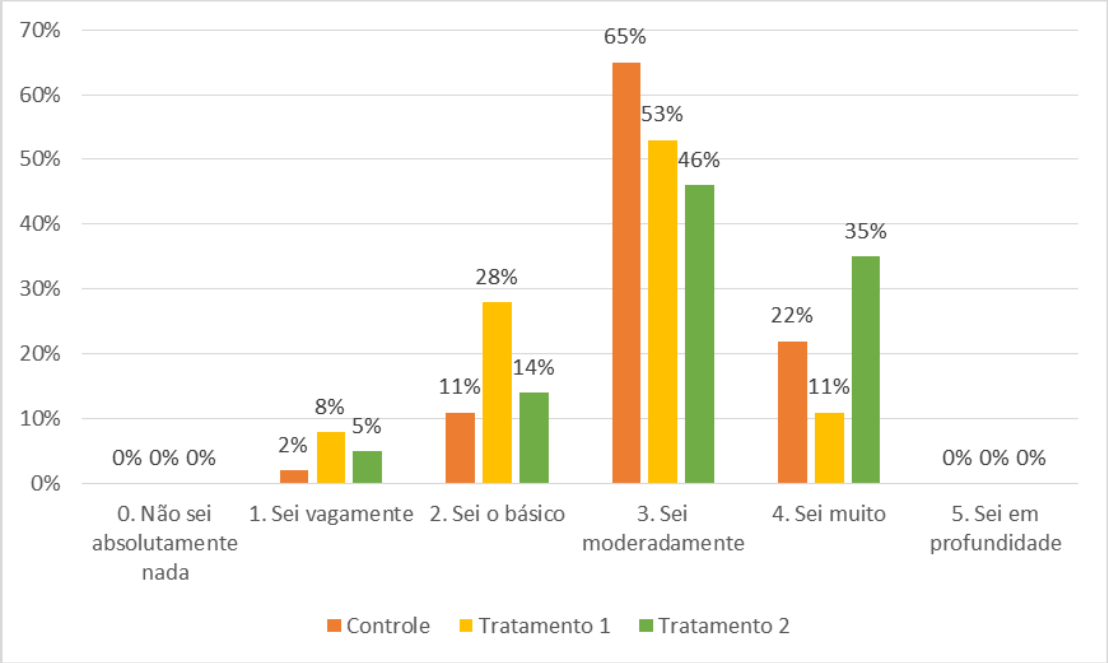
Gráfico 4 - Percentual de Graus de Aprendizagens dos Alunos (Pré-Experimento)



Observa-se que a maioria dos estudantes respondeu “Sei o básico” os tópicos de ES, prevalecendo o Grupo Tratamento 2 (50%). Adicionalmente, destaca-se a quantidade do percentual do grau “Sei moderadamente”, com ênfase para o Grupo Controle (47%).

Já o Gráfico 5 apresenta o percentual de respostas obtidas para cada um dos graus de aprendizagem, segundo os alunos dos 3 grupos após a realização do experimento.

Gráfico 5 - Percentual de Graus de Aprendizagens dos Alunos (Pós-Experimento)



Observa-se que a maioria dos estudantes respondeu “Sei moderadamente” os tópicos de ES, prevalecendo o Grupo de Controle (65%). Adicionalmente, destaca-se o percentual do grau “Sei muito” do Grupo Tratamento 2 (35%).

No geral, em comparação com os resultados obtidos na análise pré-experimento, houve um aumento do grau “Sei moderadamente” para “Sei muito”. O principal aumento de aprendizagem foi observado no Grupo Tratamento 2, seguido do Grupo Controle.

5.2. Dados Qualitativos

A) Aprendizagem das Unidades de Conhecimento da Engenharia de Software

Inicialmente, os pesquisadores definiram um gabarito para os mapas conceituais, com base nos currículos de referência da área da SBC [2] e ACM/IEEE [3], para cada unidade de conhecimento da ES. A Figura 2 apresenta o gabarito definido para a unidade de Gerenciamento de Projetos.

Em seguida, a Figura 3 apresenta um exemplo de mapa conceitual de um aluno participante do experimento, desenvolvido antes da aplicação do Tratamento 2 (modelo de ensino iterativo) e a Figura 4 que apresenta o mapa desenvolvido por este mesmo aluno após o seguimento da abordagem humanista de ensino.

Figura 2 – Gabarito do Mapa Conceitual da Unidade de Gerenciamento de Projetos

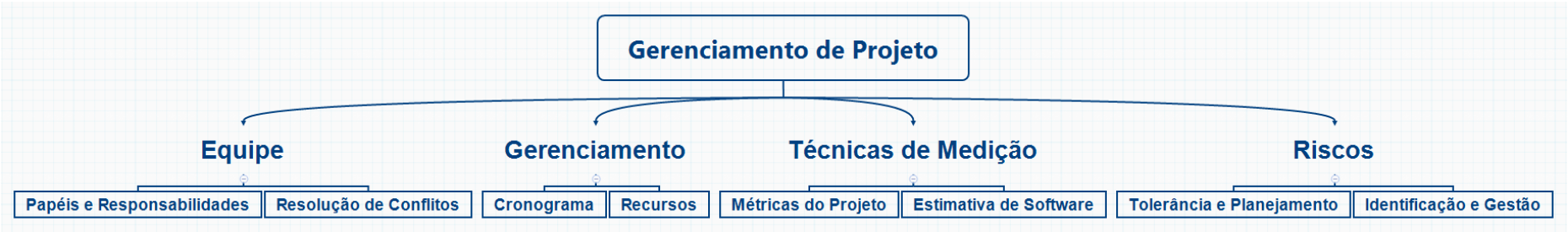


Figura 3 – Mapa Conceitual da Unidade de Gerenciamento de Projetos (Pré-Experimento)

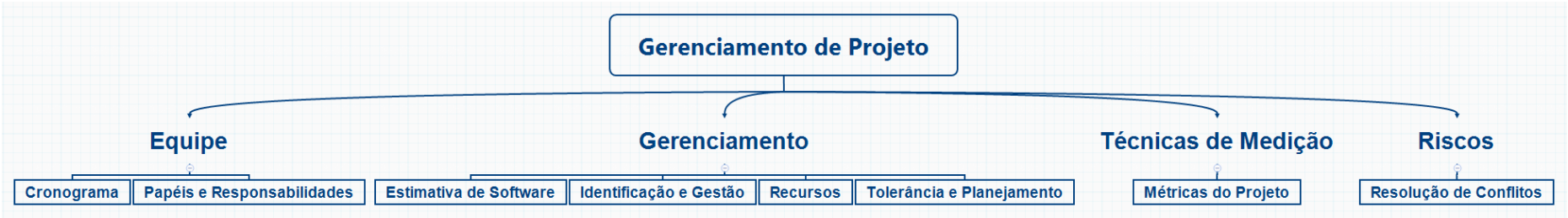
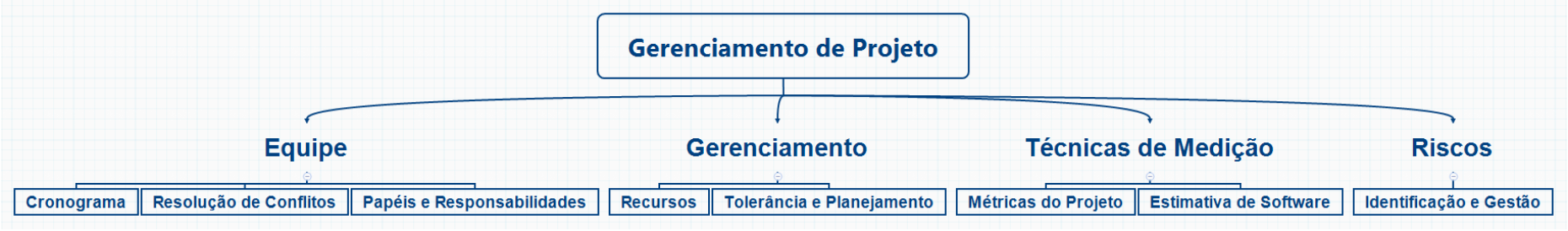


Figura 4 – Gabarito do Mapa Conceitual da Unidade de Gerenciamento de Projetos (Pós-Experimento)



Para os mapas conceituais, observa-se que na avaliação pré-experimento a menor taxa de acerto de relações entre conceitos foi 20% e a maior foi 76%. Na avaliação pós-experimento, observou-se um aumento na média de acertos, sendo a menor taxa de acerto de relações entre conceitos 53% e a maior foi 93%.

Em relação aos grupos, houve uma redução de 3% nos acertos dos alunos do Controle. Acredita-se que isso deve-se a quantidade de conteúdo teórico ministrado, que pode ter confundido as relações entre conceitos dos alunos. Não houveram alterações na quantidade de acertos dos alunos do Tratamento 1. Acredita-se que isso deve-se ao fato de que os alunos podiam consultar o mapa pré-experimento na construção do mapa pós. Devido ao baixo comprometimento deste grupo, a maioria não fez muitas alterações entre as relações definidas anteriormente. Por fim, observou-se um aumento de 9% nos acertos dos alunos do Tratamento 2. Acredita-se que esse aumento se deve ao comprometimento da equipe e o apoio do mentor e do professor (que atuou *coach* da equipe).

B) Aprendizagem dos Tópicos de Engenharia de Software

Inicialmente, os pesquisadores definiram um gabarito para os dicionários de dados, com base nos modelos de qualidade CMMI-DEV [20], MR-MPS-SW [21] e na norma ISO/IEC 12207 [22], para cada unidade de conhecimento da ES.

A Figura 5 apresenta o gabarito definido para a unidade de Gerenciamento de Projetos, a fim de compará-lo com os dicionários dos alunos pré e pós-experimento dos alunos dos 3 grupos.

Figura 5 – Gabarito do Dicionário de Dados da Unidade de Gerenciamento de Projetos

1. ***Cronograma:*** gerenciamento das datas das atividades técnicas e marcos do projeto.
2. ***Equipe:*** responsáveis pelo planejamento, gerenciamento e execução do projeto.
3. ***Estimativa de Software:*** uma técnica de medição referente ao tempo e esforço necessário para desenvolver um software específico.
4. ***Gerenciamento de Projeto:*** atividades relacionadas ao planejamento e acompanhamento do projeto de software.
5. ***Identificação e gestão:*** a identificação dos riscos pode ser feita através de uma lista a ser gerada pelo gerente do projeto e/ou equipe do projeto para identificar quais destes são potenciais riscos para o projeto em questão. A gestão consiste em executar ações de mitigação para evitar que os riscos aconteçam ou, no caso de riscos terem se concretizado, pode ser preciso executar ações de contingência para minimizar seus efeitos.

6. ***Métricas do Projeto:*** permitem acompanhar o progresso de algum estágio de um projeto usando medidas adequadas.
7. ***Papéis e responsabilidades:*** funções e atribuições que os membros da equipe devem assumir.
8. ***Recursos:*** hardware, pessoas e materiais necessários para o desenvolvimento de um projeto de software. Estes devem ser estimados e gerenciados.
9. ***Resolução de conflitos:*** compreender as fontes, perigos e benefícios potenciais de conflito na equipe.
10. ***Riscos:*** imprevistos que podem ou não ocorrer durante o projeto, podendo-o afetar negativamente ou positivamente.
11. ***Técnicas de medição:*** técnicas que permitem estimar o tamanho, duração, custo e esforço necessário para desenvolvimento de um software.
12. ***Tolerância e planejamento:*** identificar os riscos e descrever abordagens para o planejamento dos riscos (prevenção, a aceitação, a transferência, mitigação). A tolerância consiste no nível de risco aceitável.

A Figura 6 apresenta o dicionário de dados de um aluno do Grupo Tratamento 1 para a unidade de Gerenciamento de Projetos, antes de iniciar o experimento, ou seja, representando seu conhecimento prévio em relação aos tópicos de ES.

Figura 6 – Dicionário de Dados da Unidade de Gerenciamento de Projetos (Pré-Experimento)

1. ***Cronograma:*** é a descrição das atividades de trabalho de acordo com a ordem que elas devem acontecer.
2. ***Equipe:*** é um grupo de pessoas selecionadas para trabalhar em um projeto.
3. ***Estimativa de Software:*** não sei responder.
4. ***Gerenciamento de Projeto:*** não sei responder.
5. ***Identificação e gestão:*** não sei responder.
6. ***Métricas do Projeto:*** não sei responder.
7. ***Papéis e responsabilidades:*** é a função que cada pessoa tem numa equipe.
8. ***Recursos:*** são os meios utilizados para a realização de um projeto como: recursos matérias, humano e financeiro.
9. ***Resolução de conflitos:*** são as soluções para eliminar os conflitos.
10. ***Riscos:*** é uma estimativa de aceitação de um produto no mercado, por exemplo, que pode ser tanto negativa como também positiva.

11. ***Técnicas de medição:*** não sei responder.

12. Tolerância e planejamento: não sei responder.

Por fim, a Figura 7 apresenta o dicionário de dados definido por este mesmo aluno para a mesma unidade de Gerenciamento de Projetos, após a aplicação do Tratamento 1, ou seja, representando o conhecimento após o seguimento de uma abordagem iterativa de ensino focada no aluno.

Figura 7 – Dicionário de Dados da Unidade de Gerenciamento de Projetos (Pós-Experimento)

1. ***Cronograma:*** é um documento que descreve quando as atividades de trabalho devem acontecer e define o responsável por cada atividade.

2. ***Equipe:*** é um grupo de pessoas selecionadas para trabalhar em um projeto.

3. ***Estimativa de Software:*** é uma técnica utilizada no gerenciamento de um projeto que tem como objetivo estimar tempo, custo e riscos do mesmo.

4. ***Gerenciamento de Projeto:*** é a aplicação de conhecimentos e habilidades para execução de um projeto de forma ágil e eficiente.

5. ***Identificação e gestão:*** é a fase em que se identificam os riscos em um projeto e planeja as soluções.

6. ***Métricas do Projeto:*** é uma definição que descreve um atributo do projeto e que fornece maior compreensão do mesmo.

7. ***Papéis e responsabilidades:*** descreve a função e as tarefas de cada membro de um projeto.

8. ***Recursos:*** são os meios utilizados para a realização de um projeto como: recursos materiais, humano e financeiro.

9. ***Resolução de conflitos:*** são as soluções para eliminar os conflitos.

10. ***Riscos:*** é tudo que pode impactar um projeto negativamente ou positivamente.

11. ***Técnicas de medição:*** são métodos utilizados para fazer estimativas de um projeto.

12. ***Tolerância e planejamento:*** é uma estratégia utilizada por gestores de projetos para prevenir e resolver conflitos.

Para os dicionários de dados, observa-se que na avaliação pré-experimento a menor taxa de acerto de relações entre conceitos foi 14% e a maior foi 63%. Na avaliação pós-experimento, observou-se um aumento na menor taxa de acerto de relações entre conceitos subiu para 27% e a maior aumentou para 89%.

C) Desenvolvimento de Competências Técnicas em Engenharia de Software

Conforme citado na Seção 3-C, a avaliação de competências técnicas em ES foi baseada no modelo SWECOM [16]. Este modelo propõe que essas competências sejam avaliadas através de entrevistas estruturadas, onde um especialista busca identificar os *gaps* (lacunas) em uma determinada unidade de conhecimento da ES. Sendo assim, definiu-se um formulário para auxiliar a condução da entrevista estruturada.

Esta avaliação foi realizada no início e no final do experimento, a fim de medir o desenvolvimento das competências nos alunos dos 3 grupos. A Tabela 2 apresenta um exemplo formulário usado na entrevista do experimento de Gerenciamento de Projetos.

Tabela 2 – Formulário para Condução da Entrevista de Análise de Competências

Individual Gap Analysis Worksheet			
Date Completed:	11/05/2017		
Gap Analysis for:	Gerenciamento de Projeto		
SKILLS			
Participação da equipe	**HAVE 4**	**NEED 4**	**GAP 0**
Discutir comportamentos comuns que contribuem para o bom funcionamento de uma equipe.	X		
Criar e seguir uma agenda para reunião de equipe.	X		
Identificar e justificar papéis necessários em uma equipe de desenvolvimento de software.	X		
Compreender as fontes, perigos e benefícios potenciais de conflito na equipe.	X		
Estimativa de esforço	**HAVE 1**	**NEED 1**	**GAP 0**
Utilizar um método ad hoc para estimar esforço de desenvolvimento de software (por exemplo, tempo) e comparar com o esforço real necessário.	X		
Gerenciamento de projeto	**HAVE 1**	**NEED 1**	**GAP 0**
Usar um processo de software particular, descrever os aspectos de um projeto que precisam ser planejados e monitorados (por exemplo, as estimativas de tamanho e esforço, um cronograma, alocação de recursos, controle de configuração, gerenciamento de mudanças, e identificação de riscos de projetos e gestão).	X		
Técnicas de medição	**HAVE 2**	**NEED 2**	**GAP 0**
Acompanhar o progresso de algum estágio de um projeto usando métricas de projeto adequadas.	X		

Comparar técnicas de tamanho e estimativa de software de custos simples.	X		
Riscos	**HAVE 3**	**NEED 3**	**GAP 0**
Identificar os riscos e descrever abordagens para a gestão dos riscos (prevenção, a aceitação, a transferência, mitigação), e caracterizar os pontos fortes e fracos de cada um.	X		
Explicar como o risco afeta as decisões no processo de desenvolvimento de software.	X		
Identificar os riscos de segurança de um sistema de software.	X		

Este formulário apresenta as competências e as habilidades relacionadas a estas. As competências e habilidades foram retiradas do currículo de referência da ACM/IEEE [3]. De acordo com o SWECOM [16], a coluna HAVE indica que todas as habilidades da competência estão no nível avaliado. No caso desta pesquisa, selecionou-se o nível Profissional de Nível Iniciante, pois este é equivalente ao nível de um aluno concluinte de um curso de graduação [16]. Já a coluna GAP indica a lacuna de habilidades necessárias para contemplar uma competência. Por fim, a coluna NEED inclui o número de habilidades que devem ser contempladas para alcançar o nível.

Nas entrevistas pré-experimento, a menor média percentual de competência foi 18% e a maior foi 64%. Após a conclusão do experimento, a menor média foi 55% e alguns alunos alcançaram 100% das competências da unidade de Gerenciamento de Projetos.

Analisando o aumento das habilidades neste experimento, ou seja, o desenvolvimento de competências em Gerenciamento de Projetos, observa-se que o Grupo Controle obteve um aumento de 14%, o Grupo Tratamento 1 aumentou as competências em 45% e o Grupo Tratamento 2 obteve 61% de aumento. No entanto, esses resultados não foram repetidos no experimento da unidade de Engenharia de Requisitos, conforme discute a seção a seguir.

5.3. Análise das Hipóteses

A Análise de Variância (ANOVA) é um procedimento utilizado para comparar tratamentos. Um tratamento é uma condição imposta que se deseja medir ou avaliar em um experimento. Os tratamentos são chamados de variáveis independentes. Quando, em um experimento estamos interessados em estudar apenas um tipo de variável independente, dizemos que possuímos apenas um fator. Assim, neste experimento optou-

se por isolar dois tratamentos: abordagens focadas no aluno e o modelo iterativo de ensino, que além das abordagens focadas no aluno integram práticas de capacitação da indústria.

As unidades experimentais podem ser formadas por grupos ou indivíduos, por exemplo, uma turma de alunos. O uso de grupos ou indivíduos como unidades experimentais depende do fenômeno que se está estudando, da forma como o experimento é conduzido e dos recursos disponíveis. De modo geral, a escolha da unidade experimental deve ser feita de forma a minimizar o erro experimental. Por este motivo, optou-se por dividir os alunos de acordo com as avaliações pré-experimento, de forma a equilibrar os níveis dos grupos.

Uma variável é qualquer característica que apresenta variação, por exemplo, a competência dos alunos. Quando o valor de uma variável não pode ser determinado antes da realização de um experimento, tem-se então uma variável aleatória. As variáveis que assumem valores enumeráveis, são denominadas variáveis aleatórias discretas. Por exemplo, o número de alunos de uma turma. As variáveis que assumem valores em um intervalo, são denominadas variáveis aleatórias contínuas. Por exemplo, a aprendizagem dos alunos.

A partir deste procedimento de análise de dados, foi feita a verificação das hipóteses testadas, através da análise estatística das 2 fases do experimento, conforme apresenta as subseções a seguir.

A) Análise Estatística

Realizou-se a análise dos dados coletados pré e pós-experimento utilizando análise estatística. Assim, para avaliar a diferença entre os resultados de um mesmo instrumento pré e pós-experimento, utilizou-se o Teste T de *Student*, recomendado quando a estatística de teste segue uma distribuição normal, mas a variância da população é desconhecida. Nesse caso, é usada a variância amostral e, com esse ajuste, a estatística de teste passa a seguir uma distribuição T de *Student*.

Na Fase 1 do Experimento, conforme mostra a Tabela 3, o valor de P para a diferença dos dicionários de dados, mapas conceituais pré e pós-experimento foi maior que 0,05, ou seja, não houve diferença significativa. Quanto às entrevistas, o valor de P em todos os grupos foi menor que 0,05. Assim, pode-se afirmar que houve

desenvolvimento de competências, independente da abordagem (Controle, Tratamento 1 e Tratamento2).

Tabela 3 – Análise dos Resultados do Experimento Controlado 1

Instrumento	Controle	Tratamento 1	Tratamento 2
Dicionário de Dados	P = 0,26603	P = 0,15407	P = 0,17528
Mapa Conceitual	P = 0,39100	P = 0,82400	P = 0,10272
Entrevista	P = 0,01384	P = 0,00581	P = 0,00567

Na Fase 2 do Experimento, conforme mostra a Tabela 4, o valor de P para a diferença dos dicionários de dados, mapas conceituais foi menor que 0,05. Assim, pode-se afirmar que houve diferença significativa de ganho de aprendizagem de conceitos, independente da abordagem (Controle, Tratamento 1 e Tratamento2). Quanto às entrevistas, apenas o Grupo C apresentou uma diferença significativa, ou seja, houve impacto do tratamento 2 (práticas de capacitação da indústria) no desenvolvimento de competências.

Tabela 4 – Análise dos Resultados do Experimento Controlado 2

Instrumento	Controle	Tratamento 1	Tratamento 2
Dicionário de Dados	P = 0,00037	P = 0,00017	P = 0,00399
Mapa Conceitual	P = 0,01172	P = 0,01677	P = 0,03280
Entrevista	P = 0,07775	P = 0,27217	P = 0,00601

A fim de verificar se houve diferença significativa no desenvolvimento de competências técnicas de ES entre as diferentes abordagens, utilizou-se a Análise de Variância (ANOVA). Quando os resultados da ANOVA levam à rejeição da hipótese nula (abordagem humanista de ensino ≤ abordagem tradicional de ensino), que representa a afirmação de que todas as médias são iguais ou que o tratamento é menor que o controle, temos evidências de que as médias entre os níveis diferem significativamente.

A partir dessa análise, observou-se na Fase 1 do Experimento que houve diferença significativa entre os 3 grupos, sendo P = 0,00153. Adicionalmente, realizou-se análise de Teste T entre as diferentes abordagens. Os resultados obtidos ajudam a reforçar a hipótese alternativa (abordagem humanista de ensino > abordagem tradicional de ensino), pois a diferença entre o Tratamento 1 e Controle foi P = 0,01016 e entre o Tratamento 2 e Controle foi P = 0,00606. Não houve diferença significativa entre o Tratamento 1 e 2, sendo o valor de P = 0,18905.

No entanto, esses resultados não se repetiram na Fase 2 do Experimento, sendo o valor de P da ANOVA = 0,63130. Considerando que, nesta turma, o Grupo C apresentou a melhor diferença significativa (P = 0,00601) no Teste T de *Student*, pode-se inferir que o resultado em comum entre as duas fases do experimento foi o ganho significativo no desenvolvimento de competências técnicas quando se adota práticas de capacitação da indústria.

B) Teste das Hipóteses

O teste da hipótese nula (H_0) consiste em avaliar se **"**a abordagem humanista de ensino permite o desenvolvimento de competências técnicas em Engenharia de Software em nível de aplicação de maneira menos efetiva, ou igual, do que a abordagem tradicional de ensino". Nesta avaliação, o tratamento é a "abordagem humanista de ensino" e o controle é a "abordagem tradicional de ensino". Sendo assim, operacionalizou-se a variável dependente competências técnicas em Engenharia de Software a fim de medir se a variável independente abordagem de ensino possui efeito no desenvolvimento destas competências.

O fato da Fase 2 do Experimento não apontar que uma abordagem humanista possa ser mais efetiva que uma abordagem tradicional não significa que a hipótese nula é verdadeira, mas apenas que não existe evidência suficiente para confirmá-la. Nesta avaliação de uso do modelo, a eficiência dos tratamentos (abordagens focadas no aluno e práticas de capacitação da indústria) foi observada apenas na Fase 1 do Experimento da unidade de Gerenciamento de Projetos, sendo necessário algumas observações sobre a sua aplicação na Fase 2 do Experimento da unidade de Engenharia de Requisitos. Por consequência, não se pode confirmar a hipótese alternativa (H_1).

Neste sentido, destaca-se que todas as equipes tiveram a oportunidade de participar de atividades práticas, a fim de reduzir os problemas de comparação. Além disso, destaca-se que se decidiu antecipadamente quais variáveis seriam ignoradas, como por exemplo a experiência prévia do professor em abordagens humanistas e a motivação dos alunos para estudar ES, e estas podem vir a ser importantes fora do ambiente de um experimento controlado. Adicionalmente, outra limitação foi a baixa amostra da população participante do experimento: 2 professores e 30 alunos.

6. DISCUSSÕES

6.1. Avaliação dos Resultados e Implicações

O objetivo deste experimento foi "analisar o uso de uma abordagem humanista de ensino com o propósito de avaliar a sua efetividade no desenvolvimento de competências técnicas em Engenharia de Software com relação ao uso de uma abordagem tradicional de ensino". Inicialmente, definiu-se um modelo de ensino baseado nas práticas de ensino identificadas nos trabalhos de Gnatz et al. [18] e Garg e Varma [19] e Gary et al. [13]. A fim de avaliar este desenvolvimento de competências, realizaram-se entrevistas estruturadas, mapas conceituais, dicionários de dados e questionários. Os dicionários de dados criados pelos alunos permitem analisar os conceitos absorvidos por estes em relação aos tópicos de ES. Já os mapas conceituais permitem analisar como os alunos relacionam estes conceitos entre si.

Acredita-se que este objetivo foi atendido, pois todos os grupos utilizaram os mesmos instrumentos, que abordaram as mesmas unidades de conhecimento da ES. Além disso, as abordagens de ensino foram aplicadas no decorrer de um semestre pelo mesmo professor da disciplina da área de ES.

A partir da análise dos dados, onde utilizou-se análise estatística, observou-se que o grupo do Tratamento 2 teve um desenvolvimento de competência mais efetivo em relação ao grupo de Controle. No entanto, esse resultado foi observado apenas na Fase 1 do Experimento. Neste sentido, pretende-se realizar um experimento controlado de confirmação em outra instituição de ensino, a fim de reforçar as conclusões obtidas nesta pesquisa.

A partir das observações realizadas neste experimento e dos resultados obtidos, espera-se que os professores das disciplinas da área de ES possam adotar abordagens mais práticas para o ensino de ES. O caráter da disciplina de Engenharia de Software, tradicionalmente teórico, faz com que uma abordagem iterativa, onde os alunos aplicam estes conceitos em projetos de software, seja mais efetiva.

6.2. Ameaças à Validade

O fato dos dados analisados apontarem que uma abordagem humanista possa ser mais efetiva que uma abordagem tradicional não significa que a hipótese nula é verdadeira, mas apenas que não existe evidência suficiente para rejeitá-la. Neste

experimento, o valor de P foi menor que 0,05, sendo necessário algumas observações sobre estes resultados.

Para garantir que os resultados do experimento são válidos, os indivíduos foram retirados de uma amostra representativa da população. Participaram deste experimento 28 alunos e 2 professores, que representam 42,85% da população participante do *survey* [7] que motivou a realização deste experimento. No entanto, destaca-se que essa população tende a reduzir a validade externa, pois os resultados obtidos sobre esses alunos podem não se aplicar a outras instituições de ensino.

Diz-se que um experimento é válido quando é confiável e mede o que se deseja. Em outras palavras, a validade pode ser relacionada com a avaliação do experimento por algum critério externo. Neste sentido, avaliou-se a validade de constructo deste experimento. A validade de constructo consiste em quão bem um instrumento mede o constructo que se destina a medir.

Esta validade divergente do constructo pode ser avaliada por meio da correlação de um novo instrumento com um instrumento já validado. Neste caso, analisamos a validade do constructo, através da correlação do questionário deste experimento com o questionário do *survey* de Portela, Vasconcelos e Oliveira [7], que por sua vez, é baseado em outras pesquisas consolidadas na área de ensino de ES.

7. CONCLUSÕES E TRABALHOS FUTUROS

Em resumo, a pesquisa realizada através deste Experimento Controlado pode contribuir com a:

- Identificação das abordagens de ensino de ES e análise das suas vantagens e desvantagens;
- Identificação, por meio da análise do grau de aprendizagem e interesse, das abordagens mais adequadas para o ensino de tópicos de ES; e
- Identificação, por meio da análise de dicionários e mapas conceituais, das abordagens de ensino mais adequadas para desenvolver competências e habilidades profissionais de ES durante a graduação;
- Avaliação, por meio da análise da *gap analysis*, das competências técnicas de uma unidade de conhecimento de ES específica.

A principal contribuição deste Experimento Controlado é fornecer à comunidade da Engenharia de Software dados empíricos relativos a eficiência de abordagens de ensino adotadas em disciplinas de ES. Espera-se que estes dados, aliadas as análises realizadas nessa pesquisa, forneçam subsídios para professores optarem no momento de definir seus planos de ensino, considerando quais métodos de ensino seriam mais adequados na condução de disciplinas da área de Engenharia de Software.

Espera-se que os alunos possam obter uma formação mais adequada, a partir da adoção de métodos de ensino que permitam além da aprendizagem o desenvolvimento de competências e habilidades em Engenharia de Software. Para os profissionais do mercado, a contribuição desta pesquisa será, a longo prazo, a inserção de profissionais mais bem preparados para atender as demandas do mercado.

Como trabalho futuro, pretende-se realizar um experimento controlado de confirmação na Universidade Federal Rural da Amazônia (UFRA). Este experimento já está sendo planejado para o primeiro semestre de 2018 através do protocolo do experimento descrito neste documento.

REFERÊNCIAS

[1] SANTOS, R. et al. Ferramentas, Métodos e Experiências no Ensino de Engenharia de Software: um Mapeamento Sistemático. Anais do III Congresso Brasileiro de Informática na Educação. 2014. p. 544-548.

[2] SBC. Currículo de Referência da SBC para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia de Computação. Porto Alegre: Sociedade Brasileira da Computação, 2005.

[3] ACM/IEEE. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York: ACM, 2013.

[4] MARQUES, M.; QUISPE, A.; OCHOA, S. *A Systematic Mapping Study on Practical Approaches to Teaching Software Engineering*. Frontiers in Education Conference. Madrid: IEEE. 2014. p. 1-8.

[5] PRIKLADNICKI, R. et al. Ensino de Engenharia de Software: Desafios, Estratégias de Ensino e Lições Aprendidas. Anais do II Fórum de Educação em Engenharia de Software. Fortaleza, 2009.

[6] WOHLIN, C. et al. (2000) "*Experimentation in software engineering: an introduction*". Kluwer Academic Publishers, Norwell, MA, USA.

[7] PORTELA, C.; VASCONCELOS, A.; OLIVEIRA, S. Análise da Relevância dos Tópicos e da Efetividade das Abordagens para o Ensino de Engenharia de Software: Resultados de um *Survey* com Professores e Alunos. Anais do VIII Fórum de Educação em Engenharia de Software (FEES). Belo Horizonte: SBC. 2015. p. 24-35.

[8] RODRIGUES, N.; ESTRELA, N. Simple Way: Ensino e Aprendizagem de Engenharia de Software Aplicada através de Ambiente e Projetos Reais. Anais do VIII Simpósio Brasileiro de Sistemas de Informação. São Paulo, 2012. p. 722-733.

[9] COUTINHO, E.; BEZERRA, C. Aplicação de Técnicas e Conceitos no Ensino de Engenharia de Software na Faculdade Lourenço Filho. Revista Científica da Faculdade Lourenço Filho, Fortaleza, v. 7, n. 1, p. 21-36, 2010.

[10] MONSALVE, E.; WERNECK, V.; LEITE, J. SimulES-W: Um Jogo para o Ensino de Engenharia de Software. Anais do III Fórum de Ensino de Engenharia de Software (FEES). Salvador, 2010.

[11] FEITOSA, A.; CAMPOS, G. AprendES: um jogo educacional para auxiliar o processo de ensino-aprendizagem da Engenharia de Software. Anais do XXI Simpósio Brasileiro de Informática na Educação (SBIE). João Pessoa, 2010.

[12] BESSA, B.; CUNHA, M.; FURTADO, F. ENGSOFT: Ferramenta para Simulação de Ambientes Reais para auxiliar o Aprendizado Baseado em Problemas (PBL) no Ensino de Engenharia de Software. Anais do XX Workshop sobre Educação em Informática. Curitiba, 2012.

[13] GARY, K. et al. *A Project Spine for Software Engineering Curricular Design*. Proceedings of 26th Conference on Software Engineering Education and Training. San Francisco: IEEE. 2013. p. 299-303.

[14] MIZUKAMI, M. Ensino: As Abordagens do Processo. São Paulo: Epu, 1986.

[15] EASTERBROOK, S. et al. *Selecting Empirical Methods for Software Engineering Research*. In: SHULL, F.; SINGER, J.; SJøBERG, D. Guide to Advanced Empirical Software Engineering. Springer, 2007. Cap. 11.

[16] IEEE COMPUTER SOCIETY. SWECOM - *Software Engineering Competency Model*. Version 1.0. IEEE. Washington DC, p. 168. 2014. (ISBN-10: 0-7695-5373-7).

[17] JEDLITSCHKA, A.; CIOLKOWSKI, M.; PFAHL, D. *Reporting Experiments in Software Engineering*. In: SHULL, F.; SINGER, J.; SJøBERG, D. Guide to Advanced Topics in Empirical Software Engineering. Springer, 2007. Cap. 8, p. 201-228.

[18] GNATZ, M. et al. *A Practical Approach of Teaching Software Engineering*. Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03). Madrid: IEEE. 2003. p. 120-128.

[19] GARG, K.; VARMA, V. *Software Engineering Education in India: Issues and Challenges*. Proceedings of 21st Conference on Software Engineering Education and Training, Charleston, 2008. 110-117.

[20] SEI – Software Engineering Institute (2010) "*CMMI for Development* – V 1.3", http://www.sei.cmu.edu/reports/10tr033.pdf.

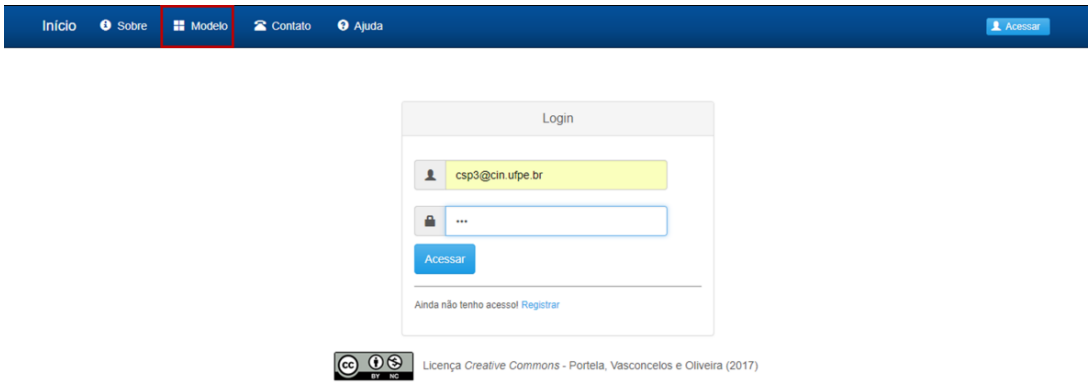
[21] SOFTEX – Associação para Promoção da Excelência do Software Brasileiro (2012) "Guia Geral MPS de Software: 2012", http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Software_20121.pdf.

[22] ISO/IEC (2008) "ISO/IEC 12207:2008 - *Systems and Software Engineering – Software Lifecycle Process*", International Organization for Standardization and the International Electrotechnical Commission, Geneva, Switzerland.

APÊNDICE G – EXEMPLO DE USO DO MODELO DE ENSINO

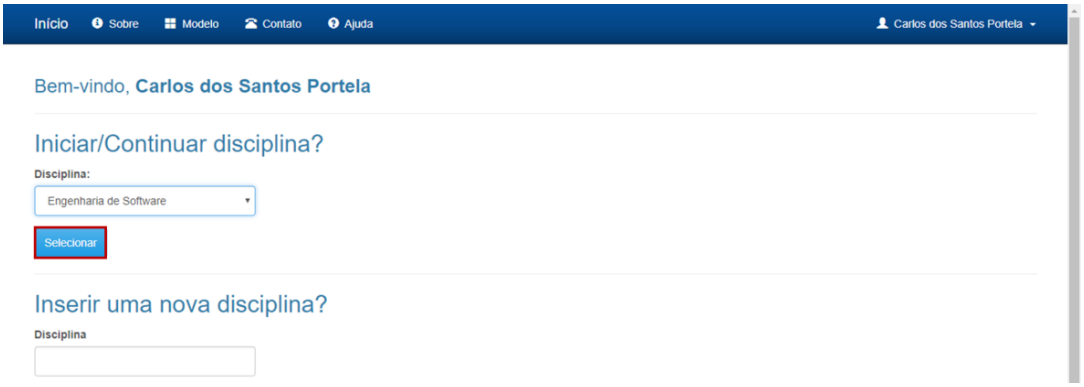
APRESENTAÇÃO

Antes de iniciar o uso do modelo de ensino, recomenda-se que o professor leia integralmente a documentação e acesse o modelo em *http://carlosportela.com.br/lafoca/*. Após acessar, efetue o *login* e clique no menu modelo, conforme Tela 1.



Tela 1 – Acessando o Modelo

Em seguida, selecione a disciplina alvo da aplicação do modelo. O professor poderá acessar uma disciplina já cadastrada ou inserir uma nova, conforme Tela 2.



Tela 2 – Inserindo uma Disciplina

Após selecionar a disciplina em que o modelo será aplicado, o professor deve determinar qual unidade de conhecimento da Engenharia de Software (ES) será o foco da aplicação. Neste exemplo de uso será adotada como referência a unidade de Engenharia de Requisitos (ER), conforme Tela 3.

Tela 3 – Selecionando uma Unidade de Conhecimento

Ao selecionar a unidade, o professor é encaminhado para o desenho da sintaxe do modelo iterativo, conforme Tela 4.

Tela 4 – Visualizando o Modelo de Ensino

Nesta tela, o professor pode acessar qualquer etapa, afim de visualizar recomendações de práticas e sugestões de materiais de aula.

A seguir, são descritos os passos necessários para instanciação das etapas do modelo em sala de aula.

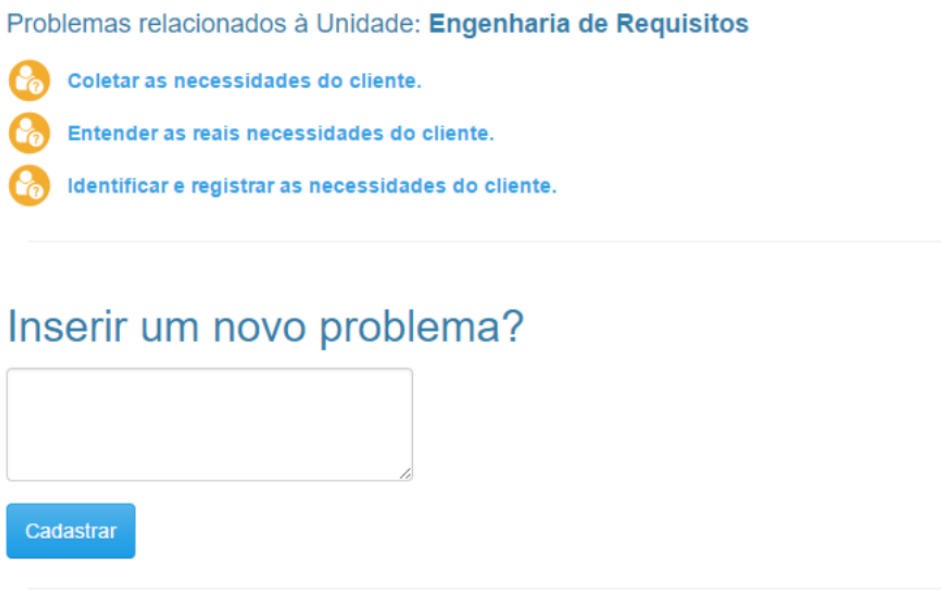
1. Iniciando o Uso do Modelo

O estudo de cada unidade de conhecimento começa a partir da identificação de um problema, relacionado ao projeto a ser desenvolvido. Essa etapa é fortemente baseada na abordagem de ensino baseada em problemas (PBL).

Antes de identificar o problema, o professor deve considerar as competências a serem desenvolvidas. De acordo com o modelo, as Competências (C) para a unidade de ER são:

- **C1**: Elicitação de Requisitos de Software;
- **C2**: Análise de Requisitos de Software;
- **C3**: Especificação de Requisitos de Software;
- **C4**: Verificação e Validação de Requisitos de Software;
- **C5**: Gerenciamento de Requisitos dos Processos e dos Produtos de Software.

Assim, baseada nestas competências e a partir da sua experiência, o professor poderia apresentar aos alunos um dos problemas apresentados na Tela 5. Caso o professor deseje, pode cadastrar um novo problema.



Tela 5 – Identificando um Problema

A partir deste problema, o professor, atuando como *coach*, poderia realizar diversas perguntas significativas, com o intuito de relacionar, implicitamente, o problema às competências a serem desenvolvidas. Alguns exemplos de Perguntas (P) que o professor pode realizar relacionadas às Competências (C) de ER são:

- **P1-C1**: Quem são os *stakeholders* (partes interessadas) que vocês consultarão para levantar os requisitos?
- **P2-C1**: Quais os métodos que vocês usarão para levantar os requisitos?

- **P1-C2**: Como vocês irão analisar se os requisitos são viáveis de serem implementados?
- **P1-C3**: Quais as notações/técnicas que vocês usarão para descrever os requisitos?
- **P1-C4**: Como vocês irão verificar se os requisitos estão claros, não ambíguos, corretos, íntegros e consistentes?
- **P1-C5**: Quais os métodos que vocês usarão para gerenciar os requisitos?

Neste sentido, essas perguntas podem ser o ponto de partida para estabelecer as metas do projeto prático. Alguns exemplos de metas (M) que o professor pode estabelecer relacionadas às competências são:

- **M1**: Identificar os *stakeholders* que atuarão como fornecedores de requisitos;
- **M2**: Selecionar os métodos para levantamento dos requisitos;
- **M3**: Analisar a viabilidade dos requisitos;
- **M4**: Selecionar as notações para especificação dos requisitos;
- **M5**: Realizar a verificação e validação dos requisitos;
- **M6**: Selecionar os métodos para gerenciamento dos requisitos.

Após selecionar a unidade de conhecimento foco da aplicação do modelo, especificar as competências que serão desenvolvidas, identificar um problema estimulante e desafiador para introduzir a unidade, realizar indagações pertinentes e estabelecer metas desafiadoras, o professor pode avançar para as etapas de Preparação e Discussão.

2. Preparando os Alunos: Fundamentação

Paralelamente a todas as etapas do ciclo iterativo do modelo, deve ocorrer a preparação dos alunos. Esta etapa consiste na realização da leitura de artigos com relato de casos práticos da indústria ou na visualização de videoaulas relacionadas à unidade.

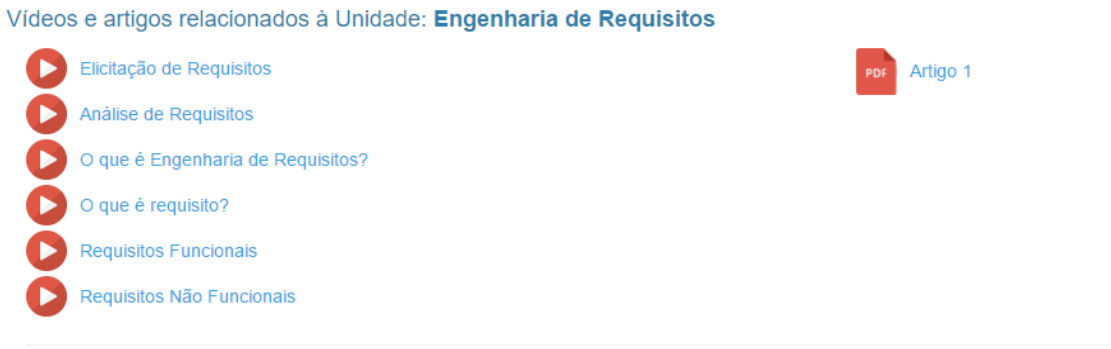
Para a unidade de Engenharia de Requisitos, o professor pode realizar a busca de artigos com relatos de experiências nos anais do Workshop em Engenharia de Requisitos (WER) da *Conferencia Iberoamericana de Software Engineering* (CIbSE) e no Simpósio Brasileiro de Engenharia de Software (SBES) do Congresso Brasileiro de Software: Teoria e Prática (CBSoft).

Quanto às videoaulas, o professor pode acessar as disponíveis no portal Videoaula@RNP, youtube (*https://www.youtube.com/*), ou qualquer outra plataforma que disponibilize esse tipo de conteúdo, preferencialmente, gratuitamente.

Por exemplo, relacionado ao P1-C2, o aluno pode acessar *http://www.videoaula.rnp.br/*, selecionar em Áreas do Conhecimento a opção *Exatas/Ciência da Computação* e depois pesquisar por "Requisitos". Assim, encontrará uma videoaula sobre Análise de Requisitos[8].

Em seguida, após obter fundamentação suficiente, o aluno pode realizar a leitura de um artigo, como por exemplo o intitulado "Uma Proposta de Elicitação e Análise de Requisitos no Contexto de Médias e Pequenas Empresas de Desenvolvimento de Software"[9] publicado no WER de 2011.

A Tela 6 apresenta as sugestões de artigos e videoaula para a unidade de Engenharia de Requisitos. Caso deseje, o professor pode cadastrar seu próprio material, seja artigo ou videoaula, ou mesmo disponibilizar algum que já tenha conhecimento e que contemple o conteúdo estudado.



Tela 6 – Acessando Vídeos e Artigos

[8] http://www.videoaula.rnp.br/portal/videoaula.action?idItem=508
[9] http://wer.inf.puc-rio.br/WERpapers/pdf_counter.lua?wer=WER11&file_name=souza.pdf

O objetivo principal é fornecer recursos para que o aluno possa adquirir fundamentação teórica, geralmente disponibilizada nas videoaulas e relatos de experiências em artigos que apresentem estudos de casos.

Esta etapa deve acontecer de forma transversal às demais, ou seja, o aluno deve realizar a preparação durante todo o processo através dos materiais recomendados pelo professor.

3. Realizando Discussão de Casos Práticos

A discussão permite aos alunos uma visão mais próxima de como determinados tópicos são aplicados na prática profissional do engenheiro de software. Além disso, permite que possam analisar criticamente e reusar as soluções técnicas.

Antes de dar início a esta fase, o professor deve convidar um profissional especialista na unidade de conhecimento. Este profissional pode ser, por exemplo, um aluno egresso da instituição que atue na indústria como Analista de Requisitos. É importante que o professor apresente o modelo de ensino ao profissional convidado e explique qual será o seu papel com relação à atividade de ensino. O professor deverá situar, ainda, o profissional quanto as metas de ensino, definidas na etapa de Iniciação, e alinhar as práticas de *mentoring* que este exercerá durante a sua interação com os alunos.

Para fins de registro, esse profissional deve ser cadastrado, conforme Tela 7.

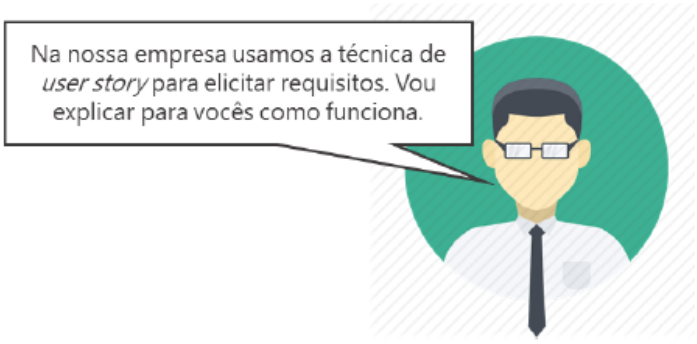
Tela 7 – Cadastrando um Profissional

Para os professores que tiverem dificuldades em conseguir um profissional que se disponibilize a participar desta experiência de ensino, uma solução consiste em convidar alunos egressos que defenderam trabalhos de conclusão de curso, ou alunos que realizaram iniciação científica com tema relacionado à unidade. Neste caso, o professor deve atentar ainda mais para as instruções referentes ao uso do modelo e ao papel que o aluno egresso deverá exercer. Recomenda-se ainda, nesta situação, que o professor o acompanhe durante a discussão e intervenha sempre que julgar necessário.

Em último caso, não havendo nem profissional nem aluno apto disponível, a discussão pode se dar a partir de uma palestra em vídeo ou videoconferência ou ainda a partir de uma visita técnica da turma a uma empresa de software. Nestas situações, o professor deverá exercer as atividades de *mentoring*, tomando como base as experiências relatadas.

Supondo que o professor tenha conseguido um profissional que atue como Analista de Requisitos, uma possível discussão poderia girar em torno do **P2-C1**, iniciada, preferencialmente, a partir de um questionamento realizado por um aluno, como por exemplo: *"Quais métodos vocês usam para levantar os requisitos?"*.

O profissional poderia responder da seguinte forma:



A partir deste questionamento, o profissional poderia contextualizar "o porquê" optaram por adotar esta técnica, como por exemplo: "*Estávamos tendo dificuldades em levantar os requisitos usando a técnica de entrevista e transcrição com determinados usuários. Com a técnica de user story (estória do usuário), o próprio usuário descreve em story cards (cartões de estória) o que espera do sistema. Como foram escritos pelos próprios usuários, o cartão pode ser usado para evidenciar o aceite dos requisitos pelos usuários*".

Adicionalmente, o profissional pode citar técnicas alternativas, métodos e ferramentas de apoio relacionadas à técnica aplicada como solução de um problema enfrentado por ele na execução de suas atividades. Estas recomendações devem ser feitas na forma de aconselhamento, conforme sugerido pelo processo de *mentoring*.

Por fim, como um mentor, espera-se que o profissional busque inspirar e motivar os alunos através de metáforas, analogias, histórias de personagens de sucesso, exemplos, demonstrações, dentre outras técnicas.

Adicionalmente, espera-se que os alunos aproveitem a oportunidade para sanar dúvidas tanto técnicas quanto referentes à atuação profissional na área. Assim, irão obter sugestões de métodos, ferramentas e técnicas que possam aplicar na etapa de Contextualização.

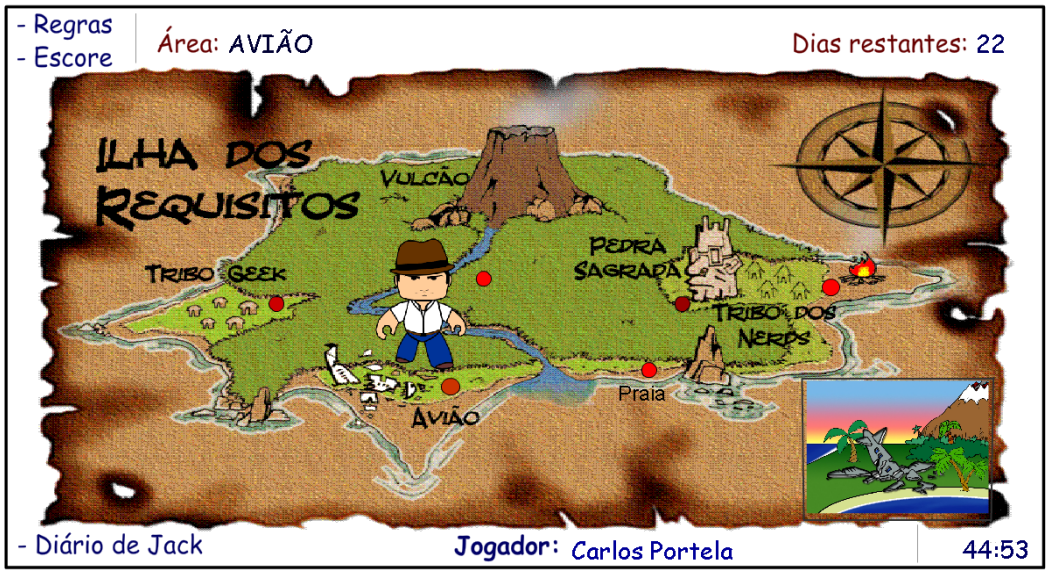
4. Realizando Atividades Práticas

Através do uso de jogos e/ou simuladores, ou da realização de workshops e/ou dinâmicas, os alunos têm a oportunidade de desenvolver experiências semelhantes às existentes no mundo real, reduzindo assim lacunas entre teoria e prática.

Antes de aplicar o conhecimento obtido na etapa de Contextualização, se faz necessário que o aluno compreenda o conteúdo estudado através da realização de atividades lúdicas. O uso de softwares educativos permite aos professores motivar e estimular os alunos quanto à aprendizagem.

Através do uso de jogos e/ou simuladores, ou da realização de *workshops* e/ou dinâmicas, os alunos podem desenvolver habilidades e apreender de maneira prática conceitos relacionados ao conteúdo da unidade de conhecimento, além de desenvolver aspectos de interação e comunicação com os demais estudantes.

Dando prosseguimento ao exemplo de uso do modelo, o professor poderia solicitar que os alunos acessem em um laboratório, atendendo aos requisitos especificados, o jogo denominado "*Ilha dos Requisitos*". Este jogo está disponível em *http://www.incremental.com.br/ilhadosrequisitos/*.



Neste jogo, o aluno/jogador se torna um sobrevivente da queda de um avião numa ilha, e deve resolver desafios referentes à unidade de ER para conseguir escapar dela. Como se trata de um jogo *single-player*, o professor pode adotar elementos da abordagem de ensino *Gamification* para motivar e engajar os alunos. Por exemplo, gerar um *ranking* a partir do desempenho obtido pelos alunos no jogo. Para o aluno que obtiver a melhor classificação, ou para os membros da equipe que obtiverem a melhor média, o professor pode recompensá-lo(s) com um exemplo de Caso de Uso modelado e descrito.

Caso prefira, o professor pode realizar a parte inicial da dinâmica "*Fábrica de Aviões*". A dinâmica completa, bem como a descrição das instruções de aplicação e avaliação dos resultados, encontra-se disponível no site da *Agile Way*[10]. O objetivo desta dinâmica é permitir aos alunos vivenciarem os problemas e desafios inerentes ao Scrum, de forma prática, facilitando a visualização de seus benefícios. Os alunos dividem-se em equipes que irão "construir" aviões de papel a partir de requisitos fornecidos pelo professor/cliente.

Nesta dinâmica, os alunos têm 3 minutos para produzir um protótipo do avião, cujos requisitos são: deve possuir 12 janelas; deve possuir uma cabine; deve possuir o símbolo da companhia aérea nas duas asas e na cauda. O que ocorre é que muitas vezes os alunos não atentam para requisitos implícitos, não especificados propositalmente pelo professor, como a porta de entrada do avião. Outras vezes, os alunos esquecem de requisitos básicos, como o símbolo na cauda. Mesmo que as vezes o protótipo contemple

[10] Disponível em http://agileway.com.br/2009/08/18/dinamica-fabrica-de-avioes-2-0/

todos os requisitos, há a possibilidade do mesmo não ser funcional, ou seja, o avião de papel não voar corretamente.

Tanto a dinâmica quanto os jogos devem ser cadastrados no site do modelo, conforme mostra a Tela 8.

Tela 8 – Cadastrando um Jogo ou Dinâmica

Além do jogo e da dinâmica, o professor pode optar pelo uso de simuladores ou pela realização de workshop sobre o uso de ferramentas de apoio, como por exemplo ferramenta de modelagem de requisitos. No entanto, recomenda-se que o professor considere o tempo disponível para o ensino da unidade no momento de decidir quantas atividades práticas irá realizar.

Após discutir e colocar em prática os conceitos internalizados, espera-se que os alunos estejam aptos a avançarem para a etapa de Contextualização, onde terão a oportunidade de realizar um projeto prático de desenvolvimento de software.

5. Contextualizando a Aprendizagem

Os alunos realizam um projeto prático de desenvolvimento de software a fim de aplicar as habilidades adquiridas. Este projeto busca a contextualização dos tópicos da unidade de conhecimento através da adoção de técnicas da Engenharia de Software.

Nesta fase, o professor deve se preocupar em fornecer ao aluno a experiência de um projeto de desenvolvimento similar ao da indústria. Neste sentido, é importante destacar que o professor deve levar em consideração as limitações do ambiente acadêmico na definição do tema, escopo, prazo e qualidade esperada para o projeto a ser desenvolvido pelos alunos.

Sendo assim, pode ser necessário selecionar quais habilidades serão priorizadas durante o projeto em detrimento de outras. Para a unidade de ER, o professor pode selecionar dentre as seguintes atividades:

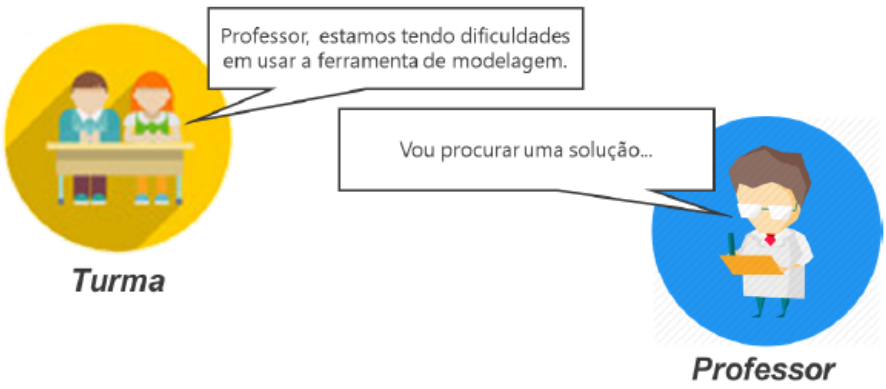
- **Elicitação**
  - Identificar os *stakeholders* (partes interessadas) para elicitação de requisitos;
  - Envolver as partes interessadas na elicitação de requisitos;
  - Usar métodos apropriados para capturar requisitos;
  - Negociar conflitos entre as partes interessadas durante a elicitação.
- **Análise**
  - Utilizar técnicas de análise de domínio apropriadas;
  - Realizar análise de viabilidade dos requisitos e propriedades emergentes.
- **Especificação**
  - Utilizar notações apropriadas para descrever requisitos.
- **Verificação e Validação**
  - Verificar os requisitos quanto à acurácia, falta de ambiguidade, integridade, consistência, rastreabilidade e outros atributos desejados;
  - Construir e analisar protótipos;
  - Negociar conflitos entre *stakeholders* durante a verificação.
- **Gerenciamento de Requisitos**
  - Usar métodos apropriados para o gerenciamento de requisitos, incluindo gerenciamento de configuração.

Estas atividades devem ser cadastradas no site do modelo, para fins de registro, conforme Tela 9.

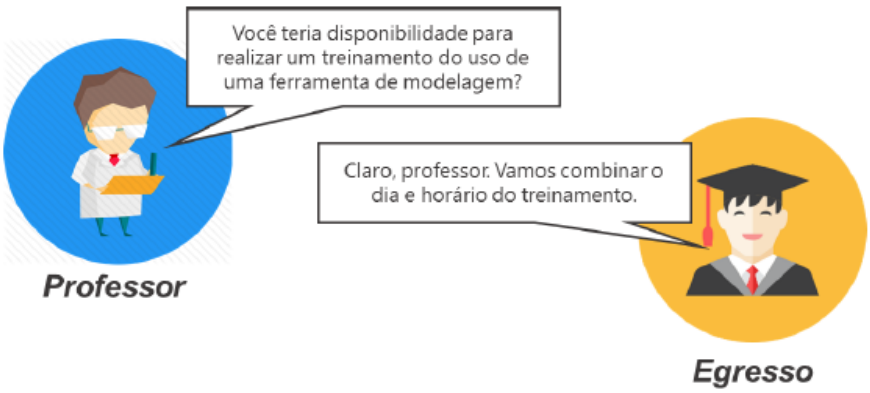
Tela 9 – Cadastrando um Projeto e Atividades

Enquanto os alunos desenvolvem o projeto, preferencialmente durante as aulas, o professor pode realizar diversas etapas do processo de *coaching*. Por exemplo: realizar indagações para auxiliar os alunos na escolha de métodos e técnicas; ressaltar as metas definidas; auxiliar na definição dos papéis e responsabilidades da equipe bem como na definição das tarefas e do cronograma; solicitar e dar *feedback* durante a realização das atividades.

O ideal é que seja estabelecido um ambiente colaborativo para o processo de ensino-aprendizagem. O professor deve atuar como *coach* dos alunos, realizando questionamentos pertinentes. Já o aluno deve, sempre que tiver dúvidas, realizar questionamentos a fim de absorver algum nível de conhecimento teórico para aplicar no projeto. Neste mesmo sentido, espera-se que eventuais problemas relativos ao desenvolvimento do projeto sejam antecipados pelos alunos.



Desta forma, os alunos podem negociar com o professor possíveis ajustes no escopo, prazo ou qualidade do projeto. O professor, por sua vez, poderá apoiar os alunos na realização das atividades, buscando soluções para estes eventuais problemas.



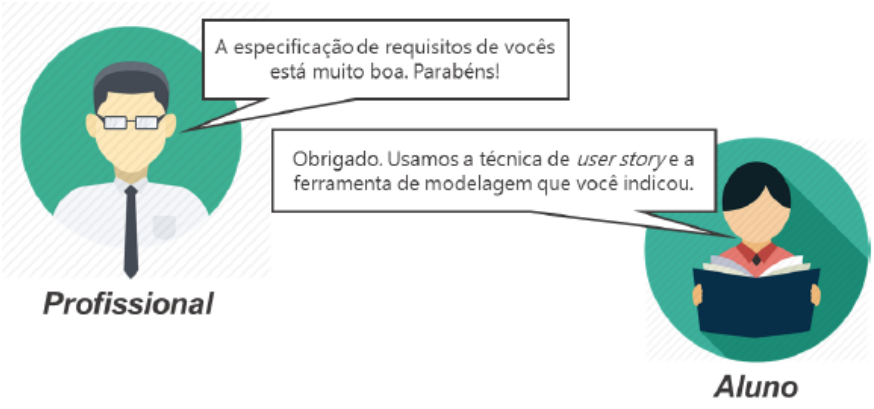
Ao finalizar as atividades práticas do projeto, solicita-se que o aluno analise o processo seguido e realize uma reflexão sobre os resultados obtidos.

6. Refletindo sobre a Aprendizagem

Os alunos devem realizar uma reflexão da experiência de aprendizagem, se envolvendo em grupos de discussão sobre as lições aprendidas. Assim, tanto os alunos quanto o professor poderão identificar e implementar melhorias para o próximo ciclo iterativo.

Esta etapa consiste numa cerimônia informal de apresentação dos resultados obtidos na etapa de Contextualização. Consequentemente, marca o encerramento do processo iterativo de ensino da unidade de conhecimento foco da aplicação do modelo. Desta forma, devem participar os alunos, o professor e demais *stakeholders* do projeto.

Se possível, o professor deve solicitar a presença do profissional convidado na etapa de Discussão. Assim, este poderá concluir o processo de *mentoring* através da visualização dos resultados da etapa de Contextualização e da análise de possíveis mudanças no comportamento de seus mentorandos.



Após a apresentação dos resultados, o professor agradece aos convidados e encerra a reunião, a fim de iniciar uma reflexão sobre o projeto com os alunos. Basicamente, os alunos deverão responder a 4 (quatro) perguntas, baseadas na cerimônia *Sprint Retrospective* do Scrum, conforme Tela 10 do site do modelo.

Tela 10 – Cadastrando as Reflexões dos Alunos

Estas reflexões devem ser armazenadas pelo professor na base de dados do sistema *web* do modelo a fim de servir de um repositório de experiências que pode auxiliar futuros alunos que irão estudar a mesma unidade de conhecimento.

O ciclo iterativo do modelo deve ser executado para cada unidade de conhecimento da ES que o professor deseja ministrar. Sendo assim, caso o professor opte por ministrar mais de uma unidade, o ciclo iterativo deve começar novamente desde a fase de Iniciação.