DOI: 10.5748/20CONTECSI/PSE/EDU/7314

eLocator: e207314

SOFTWARE ANALYSIS AND DESIGN: A COURSE PLAN USING ACTIVE TEACHING METHODS IN COMPUTER SCIENCE COURSE ANÁLISE E PROJETO DE SOFTWARE: UM PLANO DE CURSO USANDO MÉTODOS ATIVOS DE ENSINO NO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Vitor De Souza Castro – https://orcid.org/0000-0003-3209-3806
Programa De Pós-Graduação Em Ciência Da Computação (Ppgcc) / Ufpa - Universidade Federal Do Pará

Sandro Ronaldo Bezerra Oliveira – https://orcid.org/0000-0002-8929-5145
Programa De Pós-Graduação Em Ciência Da Computação (Ppgcc) / Ufpa - Universidade Federal Do Pará

SOFTWARE ANALYSIS AND DESIGN: A COURSE PLAN USING ACTIVE TEACHING METHODS IN COMPUTER SCIENCE COURSE

ABSTRACT: The teaching activity requires organization and planning of actions that will be applied in the classroom. Instruments that help the professor in this process are of extremely importance to enhance the teaching-learning process. In this sense, the objective of this work is to present a course plan for teaching Software Analysis and Design with an emphasis on active teaching methodologies. To meet this objective, the following steps were used as a methodology: literature review on the teaching approach of Software Design, mapping of competencies between the ACM / IEEE and SBC curricula for the Computer Science course related to Software Design, analysis of the main Pedagogical Course Projects of the IFES in Brazil for the Computer Science course, structuring of the course plan for teaching Software Analysis and Design and evaluation from Peer Review of the course plan. As main results, it can be highlighted the development of the course plan contemplating the main active methodologies identified in the literature, emphasis on Quality aspects in software design, indication of base literature for teaching Software Design and presentation of methodological procedures.

Keywords: Syllabus, Software Analysis and Design, Computer Science, Active Methods.

ANÁLISE E PROJETO DE SOFTWARE: UM PLANO DE CURSO USANDO MÉTODOS ATIVOS DE ENSINO NO CURSO DE CIÊNCIA DA COMPUTAÇÃO

RESUMO: A atividade docente exige organização e planejamento de ações que serão aplicadas em sala de aula. Instrumentos que auxiliam o professor nesse processo são de extrema importância para potencializar o processo de ensino-aprendizagem. Nesse sentido, o objetivo deste trabalho é apresentar um plano de curso para o ensino de Análise e Projeto de Software com ênfase em metodologias ativas de ensino. Para atender a esse objetivo foram utilizadas como metodologia as seguintes etapas: revisão da literatura sobre a abordagem de ensino de Análise e Projeto de Software; mapeamento de competências entre os currículos ACM/IEEE e SBC para o curso de Ciência da Computação relacionado a Análise e Projeto de Software; análise dos principais Projetos Pedagógicos de Curso das IFES no Brasil para o curso de Ciência da Computação; estruturação do plano de curso para o ensino de Análise e Projeto de Software; e avaliação a partir da Revisão por Pares do plano de curso. Como principais resultados pode-se destacar: o desenvolvimento do plano de curso contemplando as principais metodologias ativas identificadas na literatura, com ênfase nos aspectos de Qualidade em Análise e Projeto de Software; indicação de literatura base para o ensino de Análise e Projetos de Software e apresentação de procedimentos metodológicos.

Palavras-chave: Plano de Curso, Análise e Projeto de Software, Ciência da Computação, Métodos ativos.

Agradecimentos: Este trabalho pertence ao projeto SPIDER (*http://www.spider.ufpa.br*).

1 INTRODUÇÃO

Para um ensino eficaz, é crucial que os docentes envolvam-se no planejamento das atividades a serem aplicadas em sala de aula (Padilha, 2002). A utilização de instrumentos que apoiem os docentes nesse processo desempenha um papel fundamental na melhoria do processo de ensino-aprendizagem. Neste cenário, o plano de curso ou currículo apresenta-se como um instrumento central para que o docente tenha o direcionamento para elaboração do plano de ensino.

De acordo com Sacristán (2013), o currículo é uma seleção organizada de conteúdos a aprender, os quais, por sua vez, regularão a prática didática que se desenvolve durante a escolaridade. Lopes (2014) remete o currículo à idéia de organização, prévia ou não, de experiências/situações de aprendizagem realizadas por docente/redes de ensino de forma a conduzir o processo educativo. Logo, a definição de um currículo direciona de forma sistemática os conteúdos e as estratégias visando experiências que fortaleçam a aprendizagem dos alunos.

Para o curso de Ciência da Computação, o entendimento sobre o processo de construção de um software é exigido para formação do egresso de acordo com as diretrizes curriculares (MEC, 2016). Logo, a inclusão de disciplinas que apresentem os conteúdos sobre esse processo, considerando o processo genérico definido por Pressman e Maxim (2016) contemplando Comunicação, Planejamento, Modelagem, Construção e Entrega, torna-se fundamental para a formação do egresso.

A etapa da modelagem contempla todos os aspectos referentes ao projeto de um software. De acordo com o Bourque e Fairley (2014), a área de *Software Design* (SD) é responsável pelo processo de definição da arquitetura, dos componentes, das interfaces e de outras características de um sistema ou componente. Além disso, esta área apresenta-se como tema importante na formação do egresso de Computação (Garousi *et al*., 2019; Rodríguez-Pérez *et al*., 2021; Leite *et al*., 2020; Aniche *et al*., 2019).

Forti, Breitenbücher e Soldani (2022) apresentam que um dos desafios na área da Engenharia de Software é a alta heterogeneidade de soluções, levando a sistemas híbridos, com multi-paradigmas, sendo que esta heterogeneidade reflete em como o profissional projeta a solução, o que está diretamente associado à formação em SD.

No trabalho de Ferreira *et al.* (2018) foram identificados tópicos emergentes em engenharia de software e dentre os mais citados estão: reúso, arquitetura de software, repositórios e *domain-driven-design*. Os assuntos citados estão diretamente relacionados ao estudo de Análise e Projeto de Software (APS), o que corrobora para a motivação no desenvolvimento deste trabalho. Além disso, o mesmo trabalho de Ferreira *et al*. (2018) indica que metodologias ativas estão relacionadas com alternativas para mitigar a dificuldade dos alunos relacionada aos conteúdos lecionados na Engenharia de Software.

Neste sentido, o objetivo deste trabalho é de apresentar um plano de curso para o ensino de APS com ênfase em metodologias ativas de ensino. Para atingir este objetivo foi realizada uma revisão da literatura sobre abordagem de ensino em Projeto de Software, um mapeamento de competências entre currículos da *Association for Computing Machinery/Institute of Electrical and Electronic Engineers* (ACM/IEEE) e da Sociedade Brasileira de Computação (SBC), uma análise dos principais Projetos Pedagógicos de Curso das Instituições Federais de Ensino Superior (IFES) no Brasil, uma estruturação do plano de curso levando em consideração os resultados das etapas anteriores e uma avaliação por meio de revisão por pares.

Além desta seção introdutória, este artigo está organizado como segue: a Seção 2 apresenta o detalhamento da metodologia utilizada; a Seção 3 apresenta a proposta de plano de curso para Análise e Projeto de Software; na Seção 4 é apresentado o processo de avaliação do plano de curso; na Seção 5 são apresentadas as discussões dos resultados; na Seção 6 são apresentados alguns trabalhos relacionados; e, por fim, a Seção 7 apresenta as considerações finais, as limitações e os trabalhos futuros.

2 METODOLOGIA

Esta Seção tem o objetivo de apresentar os aspectos metodológicos para o desenvolvimento e a avaliação do plano de curso para o ensino de APS. Para tal, foi necessária a execução dos passos metodológicos, conforme Figura 1, utilizados no desenvolvimento da pesquisa, bem como são apresentados os artefatos utilizados e os artefatos produzidos por cada passo.

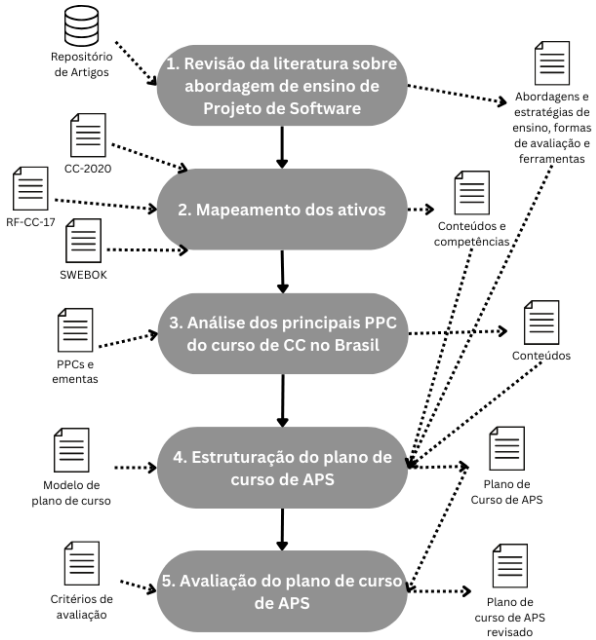

Figura 1 – Metodologia da pesquisa
Fonte**:** Elaboração própria (2023).

As subseções 2.1, 2.2 e 2.3 apresentam os passos 1, 2 e 3, respectivamente. O passo 4 é apresentado na seção 3 e o passo 5 na seção 4.

2.1 Revisão da Literatura sobre Abordagem de Ensino de Projeto de Software

No passo 1 foi conduzida uma pesquisa na literatura especializada sobre o ensino de APS (*Software Design*)contemplando: abordagem de ensino (ativa ou tradicional); forma de apresentação dos conteúdos (teórica e/ou prática); estratégias de ensino; formas de avaliação; ferramentas utilizadas; e tamanho dos grupos para atividades em equipe (Castro e Oliveira, 2023b).

Sobre o quesito abordagem de ensino, foi identificado que 90,20% dos trabalhos utilizaram a forma ativa, isso denota que a utilização de metodologias ativas de ensino deve ser aplicada para essa disciplina. Quanto à forma de apresentação dos conteúdos,

54,90% dos estudos apresentaram o ensino de forma prática e 35,29% de maneira teórica e prática, indicativo de que o ensino de forma ativa e prática apresenta-se com maior incidência na literatura especializada.

Outra questão de pesquisa do trabalho de Castro e Oliveira (2023b) foi: quais estratégias de ensino foram utilizadas no ensino de *Software Design*? Como resultados destacam-se as principais estratégias identificadas: desenvolvimento de projeto em grupo; aprendizagem baseada em projeto; aprendizagem baseada em problema; laboratório em formato *hands-on*; e sala de aula invertida. No desenvolvimento do plano de curso, Seção 3, foram utilizadas as principais estratégias de ensino identificadas na revisão.

Na revisão da literatura as formas de avaliação mais citadas foram as escolhidas para compor o plano de curso, no entanto, para melhor acompanhamento do projeto e *feedback* no processo de avaliação, optou-se por incluir entregas parciais e final do projeto, além dos tradicionais exercícios e *quizzes* individuais, que são importantes para a avaliação dos conteúdos teóricos. A apresentação oral também foi incluída como forma de avaliação, por estar relacionada, principalmente, com a estratégia de sala de aula invertida. Apesar de ser a 5ª opção de avaliação mais encontrada na revisão, o uso de prova escrita não será utilizada, pois para os conteúdos teóricos serão utilizados como forma de avaliação os exercícios e*quizzes*.

No quesito ferramentas, foram identificadas dezenas de ferramentas utilizadas no ensino de *Software Design*. Para a composição do plano de curso foi selecionado o Astah, que usa a *Unified Modeling Language* (UML) como linguagem base para desenvolvimento dos projetos de software.

Por fim, dada a maior incidência de trabalhos com abordagem prática, ativa, que utiliza estratégias de ensino em grupo, identificou-se que a composição desses grupos é de no máximo 6 alunos. Neste sentido, para as dinâmicas em grupo a abordagem definida fez o uso deste critério como limitador máximo do tamanho do grupo de alunos em determinada atividade.

2.2 Mapeamento das Competências entre os Currículos da ACM/IEEE e SBC para o Curso de Ciência da Computação

As diretrizes curriculares nacionais para os cursos de computação determinam que o perfil do formando deve enunciar as competências e habilidades desejadas (MEC, 2016). Alinhada a essa determinação, a SBC apresenta os referenciais de formação para os cursos de graduação em computação baseados em competências, sejam as genéricas, associadas ao eixo de formação, e as competências derivadas, que são relacionadas aos conteúdos específicos (Zorzo *et al*., 2017).

No contexto internacional, as entidades ACM/IEEE também definem diretrizes baseadas em competências para os cursos de computação (Zorzo *et al*., 2017)

Para a identificação das competências relacionadas aos conteúdos de APS, passo 2, foi realizado o mapeamento dos ativos entre os Referenciais de Formação da SBC para o curso de Ciência da Computação (RF-CC-17) (Zorzo *et al*., 2017), o *Computing Curricula* 2020 (CC-2020) (Curricula, 2020) e o guia do SWEBOK (Bourque e Fairley, 2014), que teve como resultado a publicação de Castro e Oliveira (2022).

O mapeamento das competências subsidiou o desenvolvimento do plano de curso de APS no direcionamento dos conteúdos e das habilidades necessárias para a formação profissional do egresso. Neste sentido, as seguintes competências dos Referenciais de Formação da SBC (Zorzo *et al*., 2017) foram selecionadas para a inclusão na proposta de plano de curso, considerando como critérios de seleção o relacionamento direto com o processo de modelagem de software e áreas adjacentes que impactam neste processo, tais como o processo de requisitos e o processo de construção de um software:

- Resolver problemas usando ambientes de programação (C1.3 e C.2.1);
- Reconhecer a importância do pensamento computacional no cotidiano e sua aplicação em circunstâncias apropriadas e em domínios diversos (C.1.5);
- Conceber soluções computacionais a partir de decisões, visando o equilíbrio de todos os fatores envolvidos (C.1.6. e C.4.7. - CE-VI);
- Aplicar temas e princípios recorrentes, como abstração, complexidade, princípio de localidade de referência (*caching*), compartilhamento de recursos, segurança, concorrência, evolução de sistemas, entre outros, e reconhecer que esses temas e princípios são fundamentais à área de Ciência da Computação (C.1.7.);

- Tomar decisões e inovar, com base no conhecimento do funcionamento e das características técnicas de hardware e da infraestrutura de software dos sistemas de computação, consciente dos aspectos éticos, legais e dos impactos ambientais decorrentes (C.2.2.);
- Avaliar criticamente projetos de sistemas de computação (C.2.3. e C.4.4. - CG-VIII);
- Empregar metodologias que visem garantir critérios de qualidade ao longo de todas as etapas de desenvolvimento de uma solução computacional (C.2.8. e C.4.8. - CE-VII);
- Analisar quanto um sistema baseado em computadores atende os critérios definidos para seu uso corrente e futuro (adequabilidade) (C.2.9. e C.3.9. - CE-VIII);
- Aplicar os princípios de interação humano-computador para avaliar e construir uma grande variedade de produtos incluindo interface do usuário, páginas WEB, sistemas multimídia e sistemas móveis (C.2.11.);
- Identificar e analisar requisitos e especificações para problemas específicos e planejar estratégias para suas soluções (C.3.8. - CE-IV).

Para a determinação dos conteúdos necessários para a disciplina de APS utilizou-se como um dos eixos a *Knowledge Area Software Design* (SD) do Guia SWEBOK (Bourque e Fairley, 2014), pois neste guia há uma estratificação de conteúdos, o que não é observado no RF-CC-17, que apresenta os conteúdos sem derivações. O SD, bem como as demais *Knowledge Areas*, possui a organização dos conteúdos em *Topics* (Tópicos) e para cada Tópico há *SubTopics* (Subtópicos) relacionados. Os Subtópicos possuem descrições que especificam quais partes desse conteúdo devem ser observadas, o que foi utilizado para as composições dos conteúdos das Unidades de Ensino da proposta do plano de curso em APS.

2.3 Análise dos Principais Projetos Pedagógicos de Curso (PPC) das IFES do Brasil para o Curso de Ciência da Computação (CC)

Para a execução do passo 3, foi realizada a análise documental dos Projetos Pedagógicos dos cursos de Ciência da Computação de 19 instituições de ensino superior com o objetivo de identificar os aspectos relacionados aos conteúdos de *Software Design* nos PPC com os resultados publicados em (Castro e Oliveira, 2023a).

Como resultados dessa pesquisa, observou-se a diversidade de disciplinas que tratam aspectos do *Software Design* e a pulverização desses conteúdos ao longo do percurso acadêmico do discente, tendo a maior concentração nos 2º e 3º anos do curso. Outro aspecto relevante foi a constatação de que os conteúdos referentes a *User Interface Design* estão concentrados nas disciplinas de Interface Humano-Computador.

Os aspectos relacionados à Qualidade do *Design* de Software são pouco explorados no cursos de CC, no entanto na proposta de plano de curso para o ensino para APS foram incluídas na Unidade III, ver seção 3.4.

A análise de correlação entre os subtópicos do SD subsidiaram o desenvolvimento das Unidades de Ensino, seção 3, e a organização dos conteúdos de maneira a tratar os assuntos com maior correlação na mesma Unidade.

A Tabela 1 apresenta os conteúdos selecionados para compor a plano de curso para o ensino de APS, sendo estes os que apresentaram maior incidência nos PPC analisados. Os critérios para a seleção dos conteúdos para o plano de curso foram: o atendimento aos principais Subtópicos identificados; e a inclusão dos Tópicos com maior incidência.

Tabela 1 – Conteúdos selecionados para compor o plano de curso

Tópico	**Subtópico**
Software Design Fundamentals	*General Design Concepts; Context of Software Design; Software Design Process; and Software Design Principles*
Software Struture and Architecture	*Architectural Strutures and Viewpoints; and*

	Families of Programs and Frameworks
Software Design Quality Analysis and Evaluation	*Quality Attributes; and Quality Analysis and Evaluation Techniques*
Software Design Strategies and Methods	*Structural Descriptions; Behavioral Descriptions; General Strategies; Object-Oriented Design; Component-Based Design; and Others Methods*
Software Design Tools	*Software Design Tools*

Os conteúdos referentes ao Tópico *Software Design Quality Analysis and Evaluation* foram incluídos no plano em função da lacuna identificada na análise dos PPC e a necessidade para o egresso do curso desta competência.

Os conteúdos relacionados ao Tópico *User Interface* não foram contemplados no plano, pois no mapeamento dos currículos e na análise dos PPC houve a concentração dos conteúdos referentes a *User Interface* em disciplinas específicas, como por exemplo Interface Humano-Computador. Além disso, esses conteúdos não estão presentes na disciplina de Engenharia de Software.

O Subtópico *Software Design Process* foi incluído, pois apresentou forte correlação com os Subtópicos do *Software Design Fundamentals*. Além disso, o *Software Design Tools* também apresentou forte correlação com os Subtópicos *Structural Descriptions* e *Behavioral Descriptions*, por isso a sua inclusão.

Por fim, a análise de correlação dos conteúdos apresentada no trabalho de Castro e Oliveira (2023a) auxiliou nas escolhas de quais conteúdos seriam incluídos nas Unidades de Ensino da proposta de plano de curso.

3 PLANO DE CURSO

Nesta Seção serão apresentados os detalhes referentes à estruturação do plano de curso para o ensino de APS. Esta estruturação utilizou como referência trabalhos que tiveram

como proposta a definição de abordagem de ensino em outras áreas da Engenharia de Software: em Testes de Software (Elgrably e Oliveira, 2022); e em Processos de Software (Quaresma e Oliveira, 2022)

Destaca-se, ainda, que o plano de curso apresentado na Seção 3.1 contemplou todos os aspectos da sua avaliação, descrita na Seção 4, sendo esta a versão final do plano de curso para o ensino de APS.

3.1Estrutura do Plano de Curso

A estrutura do plano de curso foi obtida após uma revisão informal da literatura sobre os principais componentes presentes em planos de curso e do processo de revisão por pares, Seção 4. A estrutura para a elaboração do plano para o ensino de APS, após o processo de revisão por pares, apresenta os seguintes componentes:

- Objetivo do curso: a pergunta chave para este componente é por que o curso existe (Eberly *et al*., 2001; Matejka e Kurke, 1994)?
- Pré-requisitos do curso: quais as disciplinas devem ser cursadas pelo estudante antes de realizar o curso proposto?
- Unidades de ensino: como estão organizados os conteúdos a serem apresentados para o aluno durante o curso? As Unidades de Ensino possuem os seguintes componentes:
    - Pré-requisitos: quais as disciplinas devem ser cursadas pelo estudante antes da Unidade de Ensino?
    - Questões norteadoras: perguntas que serão respondidas durante o processo de ensino e aprendizagem da Unidade;
    - Conteúdo programático: tópicos de assuntos referentes à Unidade de Ensino;
    - Estratégias de ensino: abordagem utilizada para apresentação dos conteúdos programáticos da Unidade de Ensino;
    - Estratégias de avaliação: define qual o formato de avaliação a ser utilizado na Unidade de Ensino. A estratégia de avaliação será relacionada ao procedimento metodológico;

  - Ferramentas: softwares que serão utilizados durante a execução da Unidade de Ensino;
  - Resultados esperados: o que o aluno deve ser capaz de aprender e realizar após o aprendizado da Unidade;
  - Nível de Aprendizagem: utiliza a terminologia baseada na Taxonomia Revisada de Bloom (Anderson e Krathwohl, 2001) e está relacionado às estratégias de ensino definidas na Unidade que serão especificadas em procedimentos metodológicos.
- Resultados esperados: o que o aluno deve ser capaz de aprender e realizar após o aprendizado do curso? Além disso, define quais contribuições profissionais para a formação do aluno (Bowlick *et al*., 2020).

A estrutura básica do plano de curso para o ensino de APS, de acordo com os componentes elencados nesta Seção, é apresentada na Tabela 2.

Tabela 2 – Estrutura básica do plano de curso para o ensino de APS

Objetivo do curso
Apresentar, utilizando estratégias de aprendizagem ativa, os conceitos relacionados a atividade de Análise e Projeto de Software, bem como as técnicas de projeto e requisitos de qualidade.
Pré-requisitos do curso
Engenharia de Software Banco de Dados Programação orientada a Objetos
Unidades de Ensino
I - Fundamentos de Projeto de Software

II - Estratégias para Projeto de Software

III - Qualidade em Projeto de Software

IV - Análise e Projeto orientado a objetos

Resultados esperados
O aluno deve ser capaz de compreender os conceitos relacionados a projeto de Software. Deve também situar a análise e projeto dentro do ciclo de vida do software, bem como projetar um software utilizando técnicas adequadas

Os pré-requisitos indicados na Tabela 2 remetem a conteúdos fundamentais para a compreensão do aluno neste curso. A disciplina de Engenharia de Software apresenta conteúdos gerais sobre o ciclo de vida de um software, bem como as etapas necessárias para a produção de um software. Os conteúdos de Banco de Dados e Programação Orientada a Objetos são relevantes, pois demonstram como se dará a implementação do projeto, auxiliando o aluno no processo cognitivo para se projetar um software.

As Unidades de Ensino foram criadas tendo como referência a organização dos Tópicos do guia SWEBOK (Bourque e Fairley, 2014) e a análise dos PPC das IFES, contemplando os conteúdos apresentados na Tabela 1: a Unidade I trata-se dos conteúdos referentes ao Tópico *Software Design Fundamentals*}; na Unidade II foram adicionados os conteúdos dos Tópicos *Software Struture and Architecture, Software Design Strategies and Methods*} e *Software Design Tools*; na Unidade III foram incluídos os conteúdos do Tópico *Software Design Quality Analysis and Evaluation*; e na Unidade IV foi incluído o relacionamento de todas as Unidades com o objetivo de aplicação dos conteúdos em projetos de software orientados a objetos e o desenvolvimento de documentação de arquitetura do projeto.

Nas seções 3.2, 3.3, 3.4 e 3.5 são apresentadas as Unidades de ensino contemplando os objetivos, sua estrutura de conteúdos e os principais procedimentos metodológicos da respectiva Unidade. Além disso, a Seção 3.6 apresenta as referências bibliográficas associadas por Unidade de Ensino e conteúdos, indicadas para utilização pelo docente para o desenvolvimento de materiais didáticos.

Os procedimentos metodológicos descrevem o passo a passo para a realização da prática de ensino, utilizando como base a metodologia de ensino associada. Em cada procedimento metodológico são apresentados: o tipo de atividade (individual ou grupo); a mecânica (como deve ocorrer a execução do procedimento); os critérios e as estratégias de avaliação; e o nível de aprendizagem esperado de acordo com a Taxonomia revisada de Bloom (Anderson e Krathwohl, 2001). Ao total foram especificados dez procedimentos metodológicos de ensino e a lista completa está disponível para download[1]. Neste trabalho foram incluídos quatro dos procedimentos metodológicos descritos nas Tabelas 4, 6, 8 e 10.

A seleção dos procedimentos metodológicos para compor a proposta de plano de curso ocorreu em função das estratégias de ensino com maior incidência na literatura apresentadas na Seção 2.1. Além disso, os autores deste trabalho fazem uso desses procedimentos em outras áreas do conhecimento da Computação.

Em todas as unidades de ensino foi incluída a estratégia de ensino ativa Aula expositiva dialogada, que, para Capraro (2007),é uma técnica que se propõe a substituir com vantagens a palestra docente, de modo a permitir e favorecer a participação do aluno, que traz contribuições sempre, se pertinentes ou não, à classe em conjunto com o professor. Nesta estratégia de ensino o professor tem o papel de mediador para que os alunos questionem, interpretem e discutam o objeto de estudo (Coimbra, 2017).

A relação entre os procedimentos metodológicos e os conteúdos deverão ser realizadas pelo docente na construção do plano de ensino, ou seja, quando houver a instância do plano de curso em uma turma. A proposta de seleção dos procedimentos metodológicos associados às unidades de ensino observou as características dos conteúdos da unidade.

As estratégias de avaliação estão presentes na composição de cada Unidade de Ensino e na especificação do procedimento metodológico. As estratégias de avaliação foram selecionadas em função da incidência na literatura especializada, conforme apresentada na Seção 2.1.

[1] https://anonymous.4open.science/r/sd-pm-C716/

3.2 Unidade I - Fundamentos de Projeto de Software

A Unidade I tem por objetivo apresentar os principais conceitos sobre projetos de software, bem como relembrar os alunos quanto às etapas do processo de desenvolvimento de um software e a interface entre a Engenharia de Requisitos e a etapa de APS. Por se tratar da primeira Unidade, foram adicionados em seus conteúdos assuntos de disciplinas de pré-requisitos, principalmente, a Engenharia de Software, pois apresenta de forma ampla a estrutura do processo base para o desenvolvimento de software.

A Tabela 3 apresenta a Unidade I - Fundamentos de Projeto de Software, tendo como estratégia de ensino a sala de aula invertida, pois trata-se de uma estratégia voltada para a compreensão de conceitos elementares.

Tabela 3 – Unidade I - Fundamentos de projeto de Software

Pré-requisitos
Engenharia de Software, Programação Orientada a Objetos e Algoritmos
Questões norteadoras
Quais os conceitos fundamentais envolvidos em um projeto de software? Como o projeto de software está inserido no ciclo de vida do Software ? Quais os principais conceitos relacionados à arquitetura de software ?
Conteúdos
1.1 Ciclo de vida da engenharia de Software 1.2 Engenharia de requisitos 1.3 Princípios do projeto de Software 1.4 Conceito de arquitetura de software 1.5 Tipos de projetos de software

Estratégias de Ensino	**Estratégias de Avaliação**
Sala de Aula Invertida e Aula expositiva dialogada	Apresentação Oral, Quiz e Exercícios.
Nível de Aprendizagem	**Ferramentas**
Lembrar/Conceitual Compreender/Conceitual	ChatGPT Notion
Resultados esperados	
O aluno deve ser capaz de compreender os conceitos relacionados a projeto de Software. Deve também situar a análise e projeto dentro do ciclo de vida do software.	

Sobre a estratégia de ensino definida na Tabela 3, foram desenvolvidos dois procedimentos metodológicos para a execução da Sala de Aula Invertida, sendo: Apresentação e Discussão, e *Fishbowl*.

Neste trabalho selecionou-se o *Fishbowl* como procedimento metodológico da Unidade I, definido na Tabela 4, por representar uma abordagem direcionada para o desenvolvimento de uma conversa entre os alunos que estão no círculo. Apesar da avaliação e a atividade serem caracterizadas como individuais, nota-se que a relevância da dinâmica estará na interação entre os alunos de modo a alimentar as discussões no centro do círculo.

Tabela 4 – Procedimento metodológico - *Fishbowl*

Nome	**Tipo de Atividade**
Sala de Aula Invertida: Fishbowl	Individual
Mecânica	
Disponibilizar para os alunos material sobre o tema da sala de aula invertida. Conceder prazo para leitura dos materiais e sistematização das informações usando a ferramenta Notion. Em sala, o professor deve organizar as cadeiras em círculo com 4 cadeiras no centro do	

círculo. Três alunos devem sentar no centro do círculo, deixando uma cadeira vazia.

A discussão sobre o tema da sala de aula invertida deve iniciar e no prazo de pelo menos 5 minutos para que um outro aluno possa adentrar ao centro do círculo. Assim que um aluno novo sentar, um dos alunos que estava na discussão precisa levantar-se, deixando sempre o centro com 3 alunos e 1 cadeira vazia. Este ciclo repete-se até todos os alunos participarem da discussão.

O professor deve mediar a discussão, incluindo perguntas e incentivando os alunos a participarem da dinâmica.

Avaliação	**Nível de aprendizagem**
Critérios utilizados para avaliação de cada aluno: Domínio sobre o tema; participação na dinâmica; utilização de perguntas que não foram tratadas no tema e associadas; Realização de Quiz na próxima aula sobre os conceitos e discussões realizadas; Apresentação Oral e Quiz	Lembrar/Conceitual Compreender/Conceitual

Em termos do nível de aprendizagem seguindo a definição da Taxonomia revisada de Bloom (Anderson e Krathwohl, 2001), na Unidade I o direcionamento é para os níveis da base da dimensão cognitiva, ou seja, Lembrar e Compreender.

Quanto às ferramentas, a indicação da ferramenta ChatGPT teve como propósito a inclusão de ferramentas de Inteligência Artificial para apoiar os alunos no processo de pesquisas conceituais, sendo o docente com o papel de orientar os alunos na forma de uso. Para a ferramenta Notion, a indicação foi como ferramenta de sumarização de conteúdo, colaboração *on-line* e compartilhamento.

3.3 Unidade II - Estratégias para Projetos de Software

A Unidade II - Estratégias para Projetos de Software tem como objetivo apresentar as formas de representação de um projeto de software, bem como ferramentas utilizadas neste processo e a documentação de um projeto de software. A inclusão de conteúdos sobre os tipos e a especificação de projetos fazem a Unidade II concentrar uma parte fundamental para o processo de análise de um software. Além disso, nesta Unidade será apresentada a UML, linguagem utilizada para o desenvolvimento dos artefatos de modelagem utilizadas para a documentação de um software em nível de projeto.

A Tabela 5 apresenta a estrutura da Unidade II, contemplando os conteúdos e as estratégias de ensino.

Tabela 5 – Unidade II - Estratégias para projetos de Software

Pré-requisitos
Engenharia de Software, Engenharia de Requisitos, Banco de Dados e Programação Orientada a Objetos
Questões norteadoras
Quais as estratégias utilizadas para o desenvolvimento de um projeto de software ? Como realizar a modelagem de um software ? Como realizar a especificação e documentação da arquitetura de um software ?
Conteúdos
2.1 Estratégias para projeto de software 2.2 Visões de arquitetura de software: projeto orientado a objetos, projeto de componentes, projeto de dados e projeto de interface 2.3 Ferramentas para modelagem: UML e CRC *(Class Reponsibility Card)* 2.4 Projeto baseado em padrões 2.5 Especificação da arquitetura de software: projeto web, mobile e orientado a serviço 2.6 Documentação de projeto de software

Estratégias de Ensino	**Estratégias de Avaliação**
Aula expositiva dialogada Aprendizado baseado em problema	Apresentação Oral, Quiz,Exercícios e Entrega final

ISSN 2448-1041 - TECSI – FEA USP SÃO PAULO/ BRAZIL

Laboratório *Hands-on*	
Nível de Aprendizagem	**Ferramentas**
Lembrar/Conceitual Compreender/Conceitual Aplicar/Procedural	Astah Notion
Resultados esperados	
O aluno deve ser capaz de aplicar a linguagem UML para projetar um software. O aluno deve ser capaz de compreender os principais elementos necessários para documentação sobre a etapa de modelagem de software. Além disso, utilizar ferramentas que apoiem este processo.	

Para o Unidade II o procedimento metodológico selecionado para a apresentação foi a dinâmica 24h, vide Tabela 6, que apresenta-se como a especificação da estratégia de ensino aprendizado baseado em problema. Este procedimento espera do estudante a resolução de um problema em um tempo limitado, com isso o estudante precisará de concentração e organização com a sua dupla para entregar uma solução dentro do prazo estabelecido.

Tabela 6 – Procedimento metodológico - Dinâmica 24h

Nome	**Tipo de Atividade**
Aprendizado baseado em problemas: Dinâmica 24h	Grupo com 2 alunos
Mecânica	
Realizar a divisão dos alunos em duplas e apresentação de um problema sobre um dos assuntos apresentados na Unidade. Os alunos iniciam a discussão sobre a resolução do problema em sala e caso tenham	

dúvida podem acionar o professor. A dupla deve organizar a entrega em 24h após a apresentação do problema.

Avaliação	**Nível de aprendizagem**
Critérios utilizados para avaliação de cada aluno: Nível de resolução do problema; Formatação da entrega; e Apresentação Oral;	Aplicar/Procedural

3.4 Unidade III - Qualidade em Projeto de Software

Após a apresentação dos conteúdos sobre APS, destacando o contexto da arquitetura e a sua aplicação em diferentes ambientes, a Unidade III tem como objetivo a apresentação dos principais modelos de maturidade em qualidade de software relacionados ao processo de APS, bem como indicar os atributos de qualidade associados ao projeto de software.

A Tabela 7 apresenta a estrutura de conteúdos e as estratégias de ensino que foram selecionadas para tratar sobre a temática de Qualidade em Projeto de Software.

Tabela 7 – Unidade III - Qualidade em projeto de software

Pré-requisitos
Engenharia de Software
Questões norteadoras
Quais os requisitos fundamentais para um projeto de software com qualidade ? Como um projeto de software pode ser avaliado em relação à qualidade ?
Conteúdos
3.1 Atributos de qualidade em projeto de software

3.2 Modelos de maturidade na engenharia de software 3.3 Avaliação da qualidade do projeto de software	
Estratégias de Ensino	**Estratégias de Avaliação**
Aula expositiva dialogada Aprendizado baseado em problema	Apresentação Oral, Exercícios e Entrega final.
Nível de Aprendizagem	**Ferramentas**
Analisar/Procedural Avaliar/Metacognitivo	*Checklist*
Resultados esperados	
O aluno deve ser capaz de compreender os atributos de qualidade associados ao projeto de software. O aluno deve ser capaz de realizar avaliação de qualidade em um projeto de Software.	

Os conteúdos da unidade foram encadeados para fomentar no aluno a experiência na utilização dos modelos de maturidade e a avaliação de um projeto de software considerando os atributos de qualidade.

A Unidade III foi incluída no plano de curso para atender a lacuna identificada na Subseção 2.3 que apontou que os aspectos relacionados à qualidade do *design* de software são pouco explorados no curso de Ciência da Computação.

Como uma das estratégias de ensino para a Unidade III foi projetado o procedimento metodológico Consultoria em Avaliação, vide Tabela 8. Este procedimento possibilitará aos estudantes uma experiência prática no processo de avaliação de um projeto de software. Essa dinâmica subsidiará discussões que serão realizadas na Unidade IV, onde os alunos desenvolverão um projeto de software e terão que empregar os requisitos de qualidade estudados nesta unidade.

Tabela 8 – Procedimento metodológico - Consultoria em Avaliação

Nome	**Tipo de Atividade**
Aprendizado baseado em problemas: Consultoria em Avaliação	Grupo com 6 alunos
Mecânica	
Divisão da sala em grupo com 6 alunos. Professor apresenta um problema sobre um empresa de consultoria em Qualidade de Software que precisa de um modelo de avaliação para projeto de Software. Os alunos devem produzir um modelo de avaliação com critérios/checklist para que a empresa utilize em seus trabalhos. O grupo deve apresentar a resolução do problema em sala. O professor irá medir uma discussão sobre os trabalhos e fomentar a participação dos alunos na discussão.	
Avaliação	**Nível de aprendizagem**
Critérios utilizados para avaliação do grupo: Modelo proposto; Participação na Discussão; Formatação da entrega final	Analisar/Procedural Avaliar/Metacognitivo

3.5 Unidade IV - Análise e Projeto Orientado a Objetos

A Unidade IV encerra os conteúdos referentes à Análise e Projeto de Software. Essa Unidade tem como direcionamento proporcionar ao estudante a vivência em um projeto com necessidade real, ou seja, os requisitos refletem as necessidades e vivências da organização. Além disso, essa Unidade proporcionará ao estudante atividades práticas referentes à modelagem de projeto orientado a objetos a partir do uso da UML e desenvolvimento de documentação de projeto de software.

A Tabela 9 apresenta a estrutura de conteúdos, estratégias de ensino e avaliação que serão utilizados durante esta Unidade. Destaca-se o nível de aprendizagem Criar/Metacognitivo em função da necessidade do estudante, de posse de um cenário real de problema, propor uma solução de projeto de software de forma organizada e documentada.

Tabela 9 – Unidade IV - Análise e Projeto orientado a objetos

Pré-requisitos	
Engenharia de Software, Programação orientada a Objetos e Banco de Dados	
Questões norteadoras	
Como construir um projeto de software orientado a objeto fazendo uso da modelagem UML e desenvolver a documentação consolidada do projeto ?	
Conteúdos	
4.1 Modelagem de projeto OO usando UML 4.2 Desenvolvimento de Documento de Projeto de Software	
Estratégias de Ensino	**Estratégias de Avaliação**
Aula expositiva dialogada Aprendizado baseado em projeto Laboratório *Hands-on*	Entrega parcial do projeto e Entrega final do projeto
Nível de Aprendizagem	**Ferramentas**
Criar/Metacognitivo	Astah Notion
Resultados esperados	

O aluno deve ser capaz de desenvolver e documentar um projeto de software atendendo os requisitos de qualidade, fazendo uso da UML.

Como derivação das estratégias de ensino, na Unidade IV foram previstos dois procedimentos metodológicos, sendo: Projeto orientado a objetos com UML e Projeto com necessidade real.

Para o procedimento metodológico de desenvolvimento de um projeto com necessidade real, vide Tabela 10, foi definida a atividade em grupo para o estudante ter a experiência de participar dessa etapa em equipe.

Tabela 10 – Procedimento metodológico - Projeto com necessidade real

Nome	**Tipo de Atividade**
Aprendizado baseado em projeto: Projeto com necessidade real	Grupo com 6 alunos
Mecânica	
O professor trará a demanda de um projeto de software de uma empresa local ou da própria instituição, de modo que o professor seja o ponto central das dúvidas referente ao projeto. A turma será divida em grupos com 6 alunos para atuarem no desenvolvimento projeto. Os alunos deverão produzir a documentação de projeto do software. Haverá uma entrega parcial do projeto, que será avaliada pelo professor. Ao final o grupo deve encaminhar a documentação referente a arquitetura do software fazendo uso da UML e dos requisitos de qualidades.	
Avaliação	**Nível de aprendizagem**

Critérios utilizados para avaliação de cada grupo: Qualidade dos artefatos da entrega parcial; Qualidade dos artefatos da entrega final.	Criar/Metacognitivo

3.6 Referências Bibliográficas para as Unidades de Ensino

Para auxiliar o docente no processo de elaboração de materiais didáticos referentes a cada unidade e seus conteúdos, as Tabelas 11, 12,13 e 14 apresentam referências bibliográficas, bem como a indicação de quais capítulos nessas referências estão associados a cada um dos conteúdos da proposta de curso de APS.

A Unidade I, Seção 3.2, por tratar-se de conteúdos referentes a fundamentos e princípios de projeto de software possuem referências para todos os conteúdos no livro do Pressman e Maxim (2016) e Engholm (2010)

Tabela 11 – Referências por conteúdos da Unidade I

Referência / Conteúdos	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**
Presman e Maxim (2016)	caps. 3 e 4	cap. 8	cap. 12	cap. 13	caps. 14 a 18
Sommerville (2011)	cap. 2	cap. 4	cap. 5	cap. 6	
de Pádua Paula Filho (2019)	cap. 3	cap. 4	caps. 5 e 6	cap. 6	
Engholm (2010)	cap. 2	cap. 3	caps. 3 e 4	caps. 3, 10 e 15	cap. 11
Larman (2005)	cap. 2	caps. 4 e 5	caps. 1 e 9	cap. 13	
Budgen (2003)	cap. 3		caps. 2, 5, 6 e 7		

Silveira et. al. (2011)			caps. 3 e 4	cap. 6	
Evans (2010)			caps. 1 e 2	caps. 4, 5 e 6	
Bass, Clemente e Kazman (2003)				caps. 1 e 2	

A Tabela 12 apresenta um conjunto de referências especificamente para a Unidade II, Seção 3.3. Nota-se que será necessário combinar referências para o atendimento de todos os conteúdos previstos nesta Unidade II. Destaca-se como referência bibliográfica importante para unidade: Fowler (2014), Guedes (2018) e Larman (2005), que em conjunto abordam os conteúdos referentes a UML.

Tabela 12 – Referências por conteúdos da Unidade II

Referência / Conteúdos	**2.1**	**2.2**	**2.3**	**2.4**	**2.5**	**2.6**
Presman e Maxim (2016)		caps. 13, 14 e 15	cap. 10	cap. 16	caps. 17, 18 e 19	
Sommerville (2011)		cap. 17	cap. 7	cap. 7	caps. 18 e 19	
de Pádua Paula Filho (2019)		cap. 10	caps. 2 e 9		cap. 6	cap. 6
Engholm (2010)		cap. 4	cap. 9	caps. 11 e 15	cap. 11	cap. 11
Larman (2005)		caps. 14, 17 e	caps. 6, 10, 15, 16, 28,	caps. 17, 25 e 26		caps. 22 e 39

		18	29, 37 e 39			
Budgen (2003)	caps. 5 a 9	caps. 16 e 17		cap. 10		
Evans (2010)		caps. 4 a 7		caps. 4, 5 e 6		
Bass, Clemente e Kazman (2003)				caps. 1 e 2		cap. 18
Fowler (2014)			todos caps.			
Guedes (2018)			todos caps.			

Para a Unidade III, vide Seção 3.4, as referências bibliográficas de Pressman e Maxin (2016) e Koscianski e Soares (2007) possuem todos os conteúdos que atendem a unidade. No entanto, o Guia Geral do MPS.BR para Software (SOFTEX, 2023), Chaudhary e Chopra (2016) e Chrissis, Konrad e Shrum (2011) são importantes com referências para a unidade, pois apresentam especificamente os processos e atributos de qualidade de processo para a área de APS de acordo com os modelos de maturidade MPS.BR (Melhoria do Processo de Software Brasileiro) e CMMI (*Capability Maturity Model Integration*).

Tabela 13 – Referências por conteúdos da Unidade III

Referência / Conteúdos	**3.1**	**3.2**	**3.3**
Presman e Maxim (2016)	caps. 12 e 19	cap. 21	caps. 13 e 20
Sommerville (2011)		cap. 26	

de Pádua Paula Filho (2019)		cap. 2	cap. 3
Engholm (2010)		cap. 17	
Budgen (2003)	cap. 4		
Bass, Clemente e Kazman (2003)	caps. 4 a 14		
Koscianski e Soares (2007)	cap. 11	caps. 5 e 6	cap. 12
Chaudhary e Chopra (2017)		caps. 1 e 2	
Chrissis, Konrad e Shrum (2011)		caps. 1 e parte 2	
SOFTEX (2023)		caps. 8 e 9	

Por fim, a Tabela 14 apresenta as referências que podem ser utilizadas pelo docente para o desenvolvimento dos materiais didáticos especificamente para os conteúdos da Unidade IV. Nota-se a necessidade de combinar a referência Presman e Maxim (2016) com as demais indicadas para atender aos conteúdos exigidos na Unidade IV.

Tabela 14 – Referências por conteúdos da Unidade IV

Referência / Conteúdos	**4.1**	**4.2**
Presman e Maxim (2016)		cap. 10
Sommerville (2011)	cap. 7	
de Pádua Paula Filho (2019)	caps. 2 e 9	
Engholm (2010)	cap. 9	
Larman (2005)	caps. 6, 10, 15, 16, 28, 29, 37 e 39	

Fowler (2014)	todos caps.	
Guedes (2018)	todos caps.	

4. AVALIAÇÃO DO PLANO DE CURSO

O desenvolvimento do plano de curso de APS foi submetido ao processo de revisão por pares, que trata-se de um processo colaborativo que permite com que especialistas avaliem o trabalho e forneçam sugestões e melhorias (Camargos, 2018). Além disso, Camargos (2018) aponta a rapidez, profissionalismo, colaboração e prazo como características fundamentais para o processo de revisão por pares.

O processo de revisão por pares do plano de curso de APS seguiu os passos: Identificação dos Revisores; Desenvolvimento do Formulário de Revisão; Revisão Inicial do Trabalho; Ajustes/Correção dos Itens indicados; e Revisão Final (Castro e Oliveira 2015).

O primeiro passo do processo de revisão por pares foi a identificação dos revisores. Para tal, foi encaminhado um e-mail com as informações necessárias para o processo de revisão, bem como o convite para os participantes do processo de revisão por pares. No total três avaliadores foram selecionados, e denominados de A1 - Avaliador 1, A2 - Avaliador 2 e A3 - Avaliador 3.

O processo de seleção dos avaliadores foi conduzida pelo professor orientador deste trabalho com base em sua rede de pesquisa, que envolve professores de outras instituições de ensino públicas e privadas no Brasil.

A Tabela 15 apresenta uma breve descrição do currículo dos avaliadores, bem como as áreas de atuação na engenharia de software.

Tabela 15 – Perfil dos Avaliadores selecionados no processo de Revisão por pares

Avaliador	**Perfil**

A1	Doutor e Pós-Doutor em Ciências da Computação. Avaliador e Instrutor do modelo MPS.BR de qualidade para o desenvolvimento de software. Possui experiência na área de Engenharia de Software, Qualidade de Software e Informática Educativa.
A2	Doutor em Ciência da Computação com linha de pesquisa em Engenharia de Software. Possui experiência na utilização de métodos ativos na área da Engenharia de Software.
A3	Doutor em Ciência da Computação. Possui experiência acadêmica na área de Engenharia de Software, Gerência de Projetos, Modelagem de Processos, Modelos de Qualidade do Processo de Software. Além disso, realiza pesquisas sobre o Ensino-Aprendizagem de Engenharia de Software.

O desenvolvimento do formulário de revisão utilizou como base os critérios definidos no trabalho de Castro e Oliveira (2015) contemplando os itens:

- TA - Técnico alto: indicando que foi encontrado um problema em um item que, se não for alterado, comprometerá as considerações;
- TB - Técnico baixo: indicando que foi encontrado um problema em um item que seria conveniente alterar;
- E - Editorial: indicando que foi encontrado um erro de português ou que o texto precisa ser melhorado;
- Q - Questionamento: indicando que houve dúvida quanto ao conteúdo das considerações;
- G - Geral: indicando que o comentário é geral em relação às considerações.

O formulário de revisão apresentado foi uma planilha eletrônica com duas colunas: a primeira com o critério, ou seja, para que o avaliador selecionasse - TA, TB, E, Q ou G; e

a segunda coluna para que colocasse a observação/comentário associado ao critério selecionado.

Os avaliadores selecionados foram convidados a participar de reuniões *on-line* para que fosse realizada a apresentação do plano de curso, bem como explicações sobre o formulário de revisão. Em função do número de componentes do plano de curso de APS, foram necessárias 4 reuniões para apresentação na íntegra de todos os elementos do plano de curso.

Após as apresentações, os avaliadores realizaram o preenchimento do formulário de avaliação e encaminharam para os pesquisadores. O primeiro envio dos formulários de avaliação foi considerado como o passo de Revisão inicial do trabalho. A Tabela 16 apresenta o compilado com todas as indicações realizadas pelos avaliadores na Revisão inicial do trabalho.

Tabela 16 – Consolidação dos formulários de revisão por pares

Avaliador	**Critério**	**Descrição**
A1	TA	Incluir o nível de aprendizagem da taxonomia de Bloom revisada (dimensão cognitiva e dimensão conhecimento) nos procedimentos metodológicos indicados para cada unidade especificada.
A1	TB	Remover carga horária do curso como campo da abordagem de ensino, pois não há elementos para determinar a carga horária prevista para o plano de curso.
A1	TA	Especificar a mecânica para execução dos procedimentos metodológicos definidos em cada unidade de ensino.
A2	E	Especificar os itens que são componentes na unidade de ensino de forma a deixar claro para o leitor qual o objetivo do determinado campo.

A2	E	Descrever a forma de uso das ferramentas, principalmente, o ChatGPT. Sugiro que o docente apresente para os estudantes uma forma de uso da ferramenta.
A3	TB	Remover os campos relacionados ao plano de curso da estrutura da abordagem, sendo: cronograma do curso; politica de desonestidade do aluno; política de notas para o curso; uso de tecnologias; erros comuns; como ter sucesso no curso. Indico que esses campos sejam incluídos no formato do plano de ensino (instância do plano de curso a ser aplicado quando da execução de uma experimento), dada a relevância das informações para a execução da abordagem.
A3	G	Definir em nível de capítulo as referências sugeridas por unidade de ensino.

Realizados todos os ajustes indicados na revisão inicial do trabalho, foi enviado para os avaliadores a versão final do plano de curso contemplando todas as correções indicadas na Tabela 16 Não houve solicitação de alteração na versão final demandada pelos avaliadores, encerrando o processo de revisão por pares.

Analisando os itens indicados na Tabela 16, observa-se que a maioria das indicações de ajustes na proposta foram em relação ao uso da taxonomia revisada de Bloom, citado para uso tanto na estrutura geral das unidade de ensino, quanto para os procedimentos metodológicos. Nota-se, também, que não houve indicação quanto à composição dos conteúdos contidos nas unidades, o que é um aspecto positivo no plano de curso elaborado.

5. DISCUSSÃO DOS RESULTADOS

Nesta seção serão apresentadas a análise e as discussões sobre os resultados obtidos pelo desenvolvimento do plano de curso.

O primeiro resultado importante da estruturação do plano de curso para APS é a concentração dos principais conteúdos sobre Análise e Projeto de Software, obtidos no guia SWEBOK (Bourque e Fairley, 2014) e pela análise dos PPC dos cursos de Ciência da Computação no Brasil, vide Seção 2.3, com o objetivo de proporcionar ao discente melhor formação profissional na área de computação em tópicos emergentes (Ferreira *et al*., 2018; Forti *et al*., 2022).

O uso de metodologias ativas foi incluída no plano de curso, pois trata-se de uma abordagem que é tendência no ensino e estas foram identificadas nos resultados apresentados da Seção 2.1 e nos instrumentos de diretrizes curriculares para os cursos de computação (Force, 2020; Zorzo *et al*.,2017).

Os aspectos relacionados ao projeto de interface não foram contemplados nos conteúdos de maneira detalhada no plano de curso, sendo incluídos na proposta somente apresentações gerais sobre a temática. De acordo com o trabalho de Castro e Oliveira (2023a), esses conteúdos relacionados a projeto de interface são concentrados em disciplinas exclusivas de Interação Humano-Computador.

A inclusão do procedimento metodológico "Projeto em empresa local", vide Tabela 10, busca envolver os discentes com o mercado, fazendo com que haja interação com problemas reais e apresentem uma visão de quais soluções podem ser aplicadas naquele contexto.

Por fim, a temática de qualidade, especificamente em APS, foram contempladas na proposta na Unidade III, vide Seção 3.4. Essa temática faz-se presente no guia SWEBOK (Bourque e Fairley, 2014) e ausente na maioria das disciplinas analisadas na Seção 2.3, o que motivou a sua inclusão por entender que os aspectos de qualidade são fundamentais para evolução do projeto de software.

6. TRABALHOS RELACIONADOS

Nesta seção serão apresentados alguns trabalhos relacionados bem como a diferenciação e os relacionamentos com este trabalho.

No trabalho de Quaresma e Oliveira (2022) foi desenvolvido um plano de curso para a área de Processo de Software. Além disso, para a elaboração do plano de curso foi realizada uma análise na literatura sobre os principais assuntos tratados em processo de software. Diferentemente de Quaresma e Oliveira (2022), este trabalho aborda a área de conhecimento de Análise e Projeto de Software e fez o uso de análise documental dos Projetos Pedagógicos dos cursos de Ciência da Computação do Brasil e abordagens de ensino identificadas na literatura como justificativa para inclusão no plano do curso.

No trabalho de Elgrably e Oliveira (2020) foi criado um plano de curso para o ensino de Teste de Software. Em termos de metodologia da pesquisa, há semelhança com este trabalho, no entanto quanto à estruturação das unidades de ensino, neste trabalho houve a inclusão das estratégias de ensino, estratégias de avaliação e ferramentas indicadas.

O trabalho de Bowlick, Bednarz e Goldberg (2020) trata de uma pesquisa em plano de cursos visando identificar o panorama sobre o ensino de programação de Sistemas de Informação Geográfica (GIS). Apesar de tratar sobre uma pesquisa em plano de cursos, o trabalho de Bowlick, Bednarz e Goldberg (2020) não tem o objetivo de desenvolver um plano de curso para o ensino de APS, o que difere deste trabalho.

7. CONCLUSÃO

Este trabalho apresentou uma proposta de plano de curso para o ensino de APS para o curso de Ciência da Computação utilizando metodologias ativas para o desenvolvimento dos conteúdos. Tal proposta contempla quatro unidades de ensino que se inter-relacionam para atendimento das competências necessárias para que o estudante desempenhe o seu papel como egresso do curso.

Além disso, a inclusão de uma unidade de ensino específica para Qualidade em Projeto de Software, vide Seção 3.4, atende uma lacuna existente nos PPC dos cursos de Ciência da Computação analisados, tornando esse aspecto um dos pilares para o desenvolvimento da atividade de análise e projeto de um software.

Como estratégias de ensino, em cada unidade foram adicionadas metodologias ativas diferentes, de modo a apresentar os conteúdos de diversas maneiras com o objetivo de trazer maior dinâmica no processo de ensino e aprendizagem.

Outro ponto de destaque foi o processo de avaliação do plano de curso, vide Seção 4, que indicou pontos de melhoria para a proposta de plano de curso, sendo essencial para a avaliação desta proposta por um conjunto de especialistas na área de Engenharia de Software, o que funcionou como mecanismo de mitigação de riscos do referido plano.

A ausência de trabalhos relacionados tratando do desenvolvimento de currículo para a disciplina de Análise e Projeto de Software, torna o trabalho inovador que contribui para pesquisas na área da educação em computação.

Como limitações do trabalho, os conteúdos selecionados para a composição nas unidades de ensino são exclusivos do guia SWEBOK e a análise dos PPC considerou apenas os cursos de Ciência da Computação com nota 5 do ENADE.

Para trabalhos futuros, pretende-se instanciar o plano de curso em um plano de ensino e realizar a sua execução no curso de Ciência da Computação de modo a observar os impactos no processo de ensino aprendizagem e a inclusão dos novos conteúdos na formação dos estudantes.

REFERÊNCIAS

Anderson, L. W. and Krathwohl, D. R. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Longman.

Aniche, M., Yoder, J., and Kon, F. (2019). Current challenges in practicalobject-oriented softwaredesign.In*2019IEEE/ACM41stInternationalConferenceonSoftwareEngineering: New Ideas and Emerging Results (ICSE-NIER)*, pages113–116.

Bass, L., Clements, P., and Kazman, R. (2003). *Software architecture in practice*. Addison-Wesley Professional.

Bourque, P. and Fairley, R. E., editors (2014). *SWEBOK: Guide to the SoftwareEngineering Body of Knowledge*. IEEE Computer Society, Los Alamitos, CA, 3.0edition.

Bowlick, F. J., Bednarz, S. W., and Goldberg, D. W. (2020). Course syllabi in gis programming: Trends and patterns in the integration of computer science and programming.*TheCanadianGeographer/LeGéographecanadien*,64(4):495–511.

Budgen, D. (2003). *Software design*. Addison-Wesley.

Camargos, E. F. (2018). Peer review: importance, responsibilities, and benefits. *Geriatrics, Gerontology and Aging*, 12(3):141–142.

Capraro,L.(2007).Técnicasdeensinoaserviçodoprofessorengenheiro.InXXXV CongressoBrasileirodeEducaçãoemEngenharia,COBENGE,Curitiba-PR.

Castro,V.d.S.eOliveira,S.R.B.(2015).UmframeworkdepráticasÁgeisparaapoio àimplementaçãodoprocessodeprojetoeconstruçãodoproduto.*iSys–Brazilian Journal of Information Systems*, 8(2):78–97.

Castro,V.d.S.andOliveira,S.R.B.(2022).Contentandcompetencesforteachingsoftware design in the computer science course: A mapping of CC-2020, RF-CC-2017 and SWEBOK-v3.0. *19th CONTECSI - INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS AND TECHNOLOGY MANAGEMENT VIRTUAL*.

Castro, V. d. S. and Oliveira, S. R. B. (2023a). A diagnosis on the teaching of software design in a sample of undergraduate courses in computer science in brazil. In *2023 IEEE Frontiers in Education Conference (FIE)*.

Castro,V.d.S.andOliveira,S.R.B.(2023b).Diversityinsoftwaredesignandconstruction teaching: A systematic literature review. *Education Sciences*,13(3):303.

Chaudhary, M. and Chopra, A. (2016). CMMI for development: Implementation guide.

Apress.

Chaudhary, M. and Chopra, A. (2017). *CMMI for Development*. Apress.

Chrissis, M. B., Konrad, M., and Shrum, S. (2011). CMMI for development: guidelines for process integration and product improvement. PearsonEducation.

Coimbra,C.L.(2017).Aaulaexpositivadialogadaemumaperspectivafreireana.LEAL, EdvaldaAraújo;MIRANDA,GilbertoJosé;CASANOVA,SilviaPereiradeCastro. RevolucionandoaSaladeAula:comoenvolveroestudanteaplicandotécnicasde metodologiasativasdeaprendizagem.SãoPaulo:Atlas,pages1–13.

Curricula, C. (2020). Paradigms for global computing education. *URL: https://dl.acm.org/doi/book/10.1145/3467967*.

dePáduaPaulaFilho,W.(2019).EngenhariadeSoftware:produtos.LTC,RiodeJaneiro.

de Sena Quaresma, J. A. e Oliveira, S. R. B. (2022). Evaluation and use of a student-centered syllabus for the software process subject in a postgraduate course: A quasi-experiment. Education Sciences, 12(12):851.

Eberly, M. B., Newton, S. E., and Wiggins, R. A. (2001). The syllabus as a tool for student-centered learning. *The Journal of General Education*, pages 56–74.

Elgrably, I. S. and Bezerra Oliveira, S. R. (2022). A quasi-experimental evaluation of teaching software testing in software quality assurance subject during a post-graduate computersciencecourse.*InternationalJournalofEmergingTechnologiesinLearning*, 17(5).

Elgrably, I. S. and Oliveira, S. R. B. (2020). Construction of a syllabus adhering to theteachingofsoftwaretestingusingagilepractices.In*2020IEEEFrontiersinEducation Conference (FIE)*, pages 1–9, Uppsala, Sweden.IEEE.

Engholm,H.(2010).*Engenhariadesoftwarenaprática*.NovatecEditora,SãoPaulo, Brasil.

Evans,E.(2010).Domain-DrivenDesign:Atacandoascomplexidadesnocoraçãodo software. Alta Books.

Ferreira, T., Viana, D., Fernandes, J., and Santos, R. (2018). Identifying emerging topics and difficulties in software engineering education in brazil. In *Proceedings of the*

XXXIIBrazilianSymposiumonSoftwareEngineering,pages230–239.Associationfor ComputingMachinery.

Force, C. T. (2020). *Computing Curricula 2020*. ACM.

Forti,S.,Breitenbücher,U.,andSoldani,J.(2022).Trendingtopicsinsoftwareengineering. *SIGSOFT Softw. Eng. Notes*, 47(3):20–21.

Fowler,M.(2014).UMLEssencial:umbreveguiaparalinguagempadrão.Bookman editora.

Garousi, V., Giray, G., and Tuzun, E. (2019). Understanding the knowledge gaps of software engineers: An empirical analysis based on swebok. *ACM Trans. Comput. Educ.*, 20(1).

Guedes,G.T.(2018).UML2-Umaabordagemprática.NovatecEditora.

Koscianski,A.anddosSantosSoares,M.(2007).QualidadedeSoftware-2ªEdição: Aprendaasmetodologiasetécnicasmaismodernasparaodesenvolvimentodesoftware. Novatec Editora.

Larman,C.(2005).UtilizandoUMLePadrões,3ªediçãoed.Bookman.

Leite, F. T., Coutinho, J. C. S., and de Sousa, R. R. (2020). An experience report about challenges of software engineering as a second cycle course. In *Proceedings of the XXXIVBrazilianSymposiumonSoftwareEngineering*,SBES'20,page824–833,New York, NY, USA. Association for ComputingMachinery.

Lopes, A. C. (2014). *Teorias de currículo.* CortezEditora.

Matejka, K. and Kurke, L. B. (1994). Designing a great syllabus. *College Teaching*, 42(3):115–117.

MEC(2016).Resoluçãonº05,de16denovembrode2016,diretrizescurricularesnacionaisparaoscursosdegraduaçãoemcomputação.*Technicalreport*,Ministérioda Educação-Brasil.

on Computing Curricula, J. T. F. (2013). *Computer Science Curricula 2013*.

ACM/Association for Computing Machinery.

Padilha,P.R.(2002).Planejamento dialógico:comoconstruiroprojetopolítico-pedagógicodaescola.*Cortez/InstitutoPauloFreire*.

Pressman, R. S. and Maxim, B. R. (2016). *Engenharia de software*. McGraw HillBrasil, PortoAlegre.

Quaresma, J. A. S. and Oliveira, S. R. B. (2022). A syllabus proposal for teaching of software development process in undergraduate courses in computer science. In *ProceedingsoftheXXXVIBrazilianSymposiumonSoftwareEngineering*,pages153–167.

Rodríguez-Pérez,G.,Nadri,R.,andNagappan,M.(2021).Perceiveddiversityinsoftware engineering: a systematic literature review. *Empirical Software Engineering*, 26(5):1– 38.

Sacristán,J.G.(2013).Oquesignificaocurrículo.*Sabereseincertezassobreocurrículo.*

Porto Alegre: Penso, pages 16–35.

Silveira, P., Silveira, G., Lopes, S., Moreira, G., STEPAAT, N., and Kung, F. (2011). *IntroduçãoàArquiteturadeDesigndeSoftware:UmaIntroduçãoàPlataformaJava*. Elsevier Brasil.

SOFTEX (2023). MPS.BR - Melhoria de Processo do Software Brasileiro - Guia Geral MPS de Software. Softex.

Sommerville, I. (2011). Engenharia de software. Pearson Prentice Hall, São Paulo.

Zabeu, A.C.,Rocha,A.R.,ÂngelaFilipakMachado,C.,dosSantosSouza,G.,and

Reinehr, S. (2021). *MPS.BR - Melhoria de Processo do Software Brasileiro - Guia*

Geral MPS de Software. Softex.

Zorzo, A. F., Nunes, D., Matos, E. S., Steinmacher, I., Leite, J. C., Araujo, R., Correia, R. C.M.,andMartins,S.(2017).*ReferenciaisdeFormaçãoparaosCursosdeGraduação emComputação*.SBC.