

# Transversal Integration of Generative Artificial Intelligence into Software Engineering Education: A Practice Report of the SDLC under the PBL approach.

Fabio Silva Lopes<sup>1</sup>

<sup>1</sup>Universidade Presbiteriana Mackenzie (UPM)  
Rua da Consolação, 930 – São Paulo – SP – Brasil

fabio.lobes@mackenzie.br

**Abstract.** *The rapid rise of Generative AI (GenAI) demands an urgent pedagogical shift in Software Engineering education. This paper presents an experience report on integrating GenAI across the Software Development Life Cycle within an 18-week, project-based learning curriculum for undergraduate Information Systems students. Organized into Scrum squads, students tackled a real-world demand using LLMs (e.g., ChatGPT, Google Gemini) for requirements, PlantUML modeling, prototyping, pair programming, and testing. Guided by prompt engineering and continuous code auditing, all teams achieved full project delivery within schedule—a rare outcome in previous editions—while fostering critical debates on security, technical debt, and AI ethics. The study provides a structured framework for educators teaching human-AI collaborations.*

**Keywords—** *Artificial Intelligence, Software Development Life Cycle, Software Engineering Education, Project-Based Learning, Human-AI Collaboration.*

**Resumo.** *A rápida ascensão da IA Generativa (GenAI) exige uma mudança pedagógica urgente no ensino de Engenharia de Software. Este artigo apresenta um relato de experiência sobre a integração da GenAI em todo o ciclo de vida de desenvolvimento de software, no âmbito de um currículo de 18 semanas baseado em aprendizagem por projetos para estudantes de graduação em Sistemas de Informação. Organizados em equipes Scrum, os alunos atenderam a uma demanda do mundo real utilizando LLMs (como ChatGPT e Google Gemini) para atividades de levantamento de requisitos, modelagem com PlantUML, prototipagem, programação em par e testes. Com o suporte de engenharia de prompt e auditoria contínua de código, todas as equipes concluíram o projeto integralmente dentro do prazo — um resultado raro em edições anteriores —, ao mesmo tempo em que promoveram debates críticos sobre segurança, dívida técnica e ética em IA. O estudo oferece uma estrutura organizada para educadores que ensinam sobre a colaboração entre humanos e IA.*

**Palavras-chave—** *Inteligência Artificial, Ciclo de Vida do Desenvolvimento de Software, Educação em Engenharia de Software, Aprendizado baseado em projeto, Colaboração IA-Humano.*

## 1. Introdução

A Engenharia de Software tem passado por profundas transformações desde as abordagens tradicionais do século XX até as metodologias ágeis e iterativas do século XXI, estas transformações tem sido moldadas pela necessidade contínua de gerenciar a complexidade e garantir a qualidade no desenvolvimento de sistemas de informação [Boehm, 2006]. Na busca por manter proximidade às práticas de mercado, o ensino dessa disciplina consolidou-se em torno das etapas do Ciclo de Vida de Desenvolvimento de Software (SDLC), onde os alunos aprendem a dominar o rigor de artefatos clássicos, como a elicitação de requisitos funcionais e não funcionais, matrizes de rastreabilidade, e modelagem UML (diagramas de classe, sequência e casos de uso) e praticam o desenvolvimento de aplicações com base neste aprendizado [Bordin, 2024].

Não obstante, a ascensão da Inteligência Artificial (IA), e mais especificamente da IA Generativa (GenAI), emergiu como uma força disruptiva no desenvolvimento de software. Ferramentas baseadas em Modelos de Linguagem de Grande Escala (LLMs) e sistemas agênticos passaram a atuar em praticamente todas as etapas do SDLC, desde a análise de requisitos com Processamento de Linguagem Natural (NLP), a geração de UI/UX, a codificação com *vibe code* e apoio em tempo real - com IA embutida em ferramentas como GitHub e VSCode - bem como, na automação de testes e documentação. A somatória destes fatos introduz um novo paradigma na produção de software, alterando não apenas as ferramentas utilizadas pelos desenvolvedores, mas também a própria natureza da cocriação de artefatos [Varna et al., 2023]. Ainda no contexto acadêmico, o ensino das práticas de Engenharia de Software obtém melhor eficácia quando os alunos aplicam os conhecimentos teóricos em projetos reais, utilizando abordagens como a Aprendizagem Baseada em Projetos (PBL) ou similares [Lopes; Eliseo, 2022], [Fernandez et al., 2019], [Córdova et al., 2020]. Contudo, essa disrupção tecnológica inserida de modo desregulado no ambiente educacional traz consigo novos desafios para o processo ensino-aprendizagem. A facilidade de geração de código e de artefatos de documentação por meio da IA levanta questões sobre a retenção do aprendizado fundamental, a qualidade estrutural das soluções geradas e as implicações éticas, como vieses, privacidade e propriedade intelectual. Torna-se imprescindível, portanto, que a educação em Engenharia de Software não apenas ensine a usar a ferramenta, mas fomente o pensamento crítico, a revisão rigorosa e a colaboração efetiva entre humanos e IA.

A rápida adoção de ferramentas de IA Generativa pelo mercado, aumenta a lacuna entre o ensino tradicional e as práticas de mercado, que mantém um hiato em expansão. Alguns estudos argumentam que a natureza dos problemas do mundo real e as práticas da indústria mudam rapidamente, exigindo que os programas de Engenharia de Software na academia se atualizem para incluir o desenvolvimento ágil e resolver questões técnicas, sociais e cognitivas contemporâneas [Varna et al., 2023].

Observa-se então a carência de estudos na literatura científica sobre práticas pedagógicas que abordam de modo estruturado um processo de ensino, onde o aluno revisita as bases do SDLC agregando a camada de IA Generativa de forma crítica na adoção desta abordagem.

Diante deste contexto, este trabalho investigou o seguinte problema de pesquisa: Como integrar efetivamente uma camada de Inteligência Artificial Generativa a uma disciplina prática de Engenharia de Software, de modo a revisar e fortalecer os conceitos

clássicos do SDLC, fomentando simultaneamente a proficiência técnica e o pensamento crítico dos estudantes na colaboração humano-IA.

Para responder a questão de pesquisa, o objetivo deste estudo pautou-se em relatar e avaliar a experiência de aplicação de uma camada de IA Generativa ao SDLC em uma disciplina de prática projetual em Engenharia de Software.

Este artigo relata a experiência de estruturação e execução de uma disciplina projetual com foco na revisão do SDLC por meio da implementação de um aplicativo que atende uma demanda real. Utilizou-se de modo complementar uma avaliação dos artefatos (organização, completude e rastreabilidade) utilizando também a Engenharia de Prompts e o desenvolvimento assistido por professores.

De modo empírico, a IA Generativa foi inserida em cada etapa do SDLC, por meio de ciclos de revisão da etapa, inserção da Engenharia de Prompt, prática de construção de artefatos e avaliação do artefato e análise crítica da etapa na Sprint Review, com ênfase na análise crítica dos resultados apresentados.

De modo complementar, considerou-se a investigar o desenvolvimento do pensamento crítico dos alunos ao revisar, depurar e validar os resultados gerados pelos modelos de IA.

A estrutura deste trabalho está organizada da seguinte forma: a Seção 1 introduz o tema, abordando o problema de pesquisa, os objetivos e o percurso metodológico. A Seção 2 revisa a literatura relevante para fundamentar o relato exposto na Seção 3. Os principais achados são analisados e discutidos na Seção 4. Concluindo o estudo, a Seção 5 apresenta as considerações finais e aponta perspectivas de trabalhos futuros.

## **2. Fundamentação Teórica**

Desde o surgimento da IA generativa, a preocupação com os impactos no ensino esteve presente em diversos segmentos da educação. O estudo [Ramos Da Silva et al., 2023] explorou esta preocupação de forma abrangente, e identificaram impacto substancial em temas como acesso ao conhecimento e a personalização da aprendizagem.

O framework da UNESCO sobre competências de IA para professores (Miao, Cukurova, 2024) ressalta a IA como agente viabilizador para novas formas de ensino, capazes de aprimorar as experiências de aprendizagem e apoiar as tarefas dos professores. Contudo, alerta para riscos significativos aos alunos, a comunidade docente, os sistemas educacionais e a sociedade em geral por meio da redução de processos de ensino e aprendizagem a tarefas automatizadas, que desvalorizam o papel e a influência dos professores, ou ainda, que enfraquecem a relação professor-aluno.

Nesta linha, a inserção de tecnologias de IA pode exigir uma orientação mais ativa por parte dos educadores no sentido de promover uma compreensão mais profunda e crítica. Vale salientar que a IA potencializa a aprendizagem replicando a capacidade humana de raciocinar, tomar decisões e resolver problemas, mas caberá ao professor, promover o pensamento crítico sobre o que se aprende ou se produz.

No caso do ensino em Engenharia de Software o impacto é considerável, posto que a área está vivendo uma mudança de paradigma fundamental, transitando de processos puramente executados por humanos para uma orquestração colaborativa entre humanos

e Inteligência Artificial (IA) ao longo de todo o Ciclo de Vida de Desenvolvimento de Software (SDLC).

Embora as metodologias tradicionais, como o modelo cascata e as práticas ágeis, tenham sido historicamente essenciais para gerenciar a complexidade do desenvolvimento, as crescentes demandas da indústria por ciclos de entrega cada vez mais curtos e flexíveis impulsionaram a adoção de sistemas inteligentes em toda a esteira de produção [Goel et al, 2024].

O artigo de Varna e colegas [Varna et al., 2023] explora a transição da Engenharia de Software para um paradigma de colaboração entre humanos e inteligência artificial, explorando detalhes de como a automação inteligente otimiza todo o ciclo de vida de desenvolvimento de software (SDLC), desde a extração semântica de requisitos até a síntese de código por algoritmos genéticos e o controle de qualidade automatizado.

Além de destacar os ganhos quantitativos de produtividade, os autores enfatizam a necessidade de aplicar padrões de design específicos para sistemas de aprendizado de máquina para mitigar riscos arquiteturais e dívidas técnicas.

Já o estudo (Kozub et al., 2025) ressalta que as IAs Generativas podem auxiliar analistas de negócios responsáveis por coletar e documentar requisitos. Ao implantar modelos de extração semântica, os analistas podem ingerir automaticamente narrativas de usuários, manuais legados e documentos de conformidade regulatória.

O sistema de IA realiza análise semântica automatizada, identificando contradições lógicas, classificando requisitos e filtrando documentação redundante. Estas atividades, segundo os autores podem levar à redução de 20 a 40% do tempo de desenvolvimento de código. De modo complementar, este estudo não discute o "vibe code" como ferramenta que gera 100% do código sem a participação direta do desenvolvedor.

A aplicação da IA no desenvolvimento de software exige uma análise abrangente que equilibre as descobertas práticas com as restrições sistêmicas consistentemente observadas no discurso acadêmico (Kozub et al., 2025).

Traduzir estas restrições em termos técnicos, organizacionais e legais, ainda representam fronteiras que demandam esforço por parte dos educadores na área de Engenharia de Software, pois o contexto da sala de aula tem limitações para construir estas restrições, mesmo quando lida com demandas reais.

É evidente que a demanda original de um curso de Engenharia de Software é o ensino. Portanto, questões organizacionais e legais são discutidas, mas há um limite que a sala de aula acaba por estabelecer pois não há contratos, ambiente, hierarquia real e atores presentes, entre outros limitantes.

Embora o estudo mencione que as recomendações observadas são relevantes para cenários "acadêmicos e de pesquisa", o foco do trabalho está voltado para o impacto operacional e os desafios de integração da IA na indústria.

Neste cenário, observa-se uma ampliação da distância entre práticas de mercado e o que se ensina nas universidades. Esta distância pode ser reduzida com a adoção de IA ao contexto da sala de aula. No entanto, cabe aqui uma necessidade de uma discussão mais profunda sobre como o ensino efetivo na área de Engenharia de Software será conduzido, observado a inevitável uso da IA, já adotado em muitas instâncias do mercado de trabalho.

### 3. Relato do Processo

Nesta seção vamos relatar uma prática experimental ofertada a duas turmas de graduação no terceiro ano do bacharelado em Sistemas de Informação.

Trata-se de uma disciplina projetual de Análise e Desenvolvimento de Sistemas, onde os alunos são desafiados a rever todas as etapas do SDLC aprendidas em disciplinas anteriores, para entregar uma aplicação em funcionamento, código e a respectiva documentação da especificação no repositório GitLab da escola. Todo o material documental é escrito em Markdown, e os diagramas UML são entregues no formato PlantUML.

Como fator agregador, o curso propôs o uso experimental e transversal da IA Generativa em todas as tarefas do projeto, desde a elaboração de Casos de Uso, Wireframes, Diagramas de Domínio e de Sequência, até o desenvolvimento de Front-end, Back-end e Casos de teste.

A proposta curricular foi estruturada para 18 semanas com atividades presenciais e remotas. Optou-se pela abordagem PBL para apresentar a demanda, aliada a um processo cascata para cobrir todo o SDLC, subdividido em quatro Sprints, com entregas definidas no cronograma inicial.

No primeiro semestre do experimento, o grupo de 25 alunos foi dividido em cinco Squads de igual tamanho. No segundo semestre a turma de 21 alunos foi dividida em quatro Squads, sendo que uma delas foi composta por seis alunos.

Os estudantes são organizados em Squads e cumprem um ritual Scrum de iterações quinzenais (Sprints). As Sprints foram organizadas pelos professores, que assumem o papel de Scrum Master. De modo complementar, um stakeholder real é inserido no processo para tirar as dúvidas que as Squads levantaram. Todas as Squads trabalham no mesmo projeto, com os mesmos objetivos, entregas e cronograma.

O projeto escolhido, é uma demanda real da área administrativa da faculdade. Optou-se por escolher uma demanda que os alunos não estivessem envolvidos diretamente, obrigando eles a fazer uma imersão mais robusta sobre o tema e os documentos apresentados. De modo complementar, criou-se um ambiente de gamificação, onde o melhor projeto pode ser implantado para atender à demanda identificada.

Como sugerido em (Lopes, Eliseo, 2022), a abordagem de ensino baseada em PBL corrobora com a prática aliada a uma demanda real para grupos de alunos com diferentes capacidades, onde a colaboração é chave para a resolução de um problema proposto. Nesta linha, optou-se por apresentar a demanda como ela está apresentada aos stakeholders. No caso, disponibilizou-se o Regulamento Acadêmico e o site institucional com as instruções de operação do processo. O P.O.se apresentou ao grupo e explicou o processo em uma reunião técnica para levantamento questionamentos adicionais.

Cada aula foi estruturada com três momentos distintos a saber: Primeiro momento - análise crítica sobre as tarefas executadas e artefatos desenvolvidos desde o encontro anterior (produzidos nas atividades remotas); Segundo momento, revisão do ponto do

SDLC e artefatos necessários para desenvolvimento até a semana seguinte; Terceiro momento, apresentação de estratégias para desenvolver os próximos artefatos com uso de IA generativa.

Na sequência há um resumo da produção planejada para cada sprint do projeto.

–Sprint 0 - Planejamento (Semana 1): Planejamento das atividades, divulgação do cronograma, definição das Squads e avaliação diagnóstica sobre expectativas do curso.

–Sprint 1 - Fundamentos e Introdução à IA Generativa (Semanas 2-5): Os alunos revisam os conceitos clássicos da Engenharia de Software (SDLC; metodologias ágeis; artefatos) e são introduzidos ao impacto da IA Generativa na área. As atividades incluem as primeiras interações com os stakeholders para o levantamento de requisitos dos projetos. Os alunos aprendem a usar a IA para auxiliar na estruturação e análise desses requisitos e são introduzidos aos fundamentos da Engenharia de Prompts, explorando ferramentas como Google Gemini, OpenAI ChatGPT e Anthropic Claude e Microsoft Copilot para gerar os primeiros artefatos. Outros artefatos de engenharia de software são produzidos nesta etapa. Casos de Uso, Diagramas de sequência, Diagramas de Classes de Domínios. Aqui, optou-se pela construção dos diagramas com uso do PLantUML. Desta forma, os diagramas podem ser gerados por script na IA e renderizados no GitLab.

–Sprint 2 - Desenvolvimento com IA Generativa - Front-end (Semanas 6-9): É a iteração de produção da interface da aplicação, produção de Wireframes, elaboração de mockups da aplicação. A IA é utilizada para colaborar na produção de interfaces utilizando a identidade institucional, paleta de cores, questões de acessibilidade, etc. A especificação segue com a produção de diagramas UML e validações com stakeholders. No final desta sprint as squads entregam toda a especificação do projeto em formato pré-definido pelos professores, no GitLab.

–Sprint 3 - Desenvolvimento com IA Generativa - Back-end (Semanas 10-13): É a fase de maior "mão na massa", focada em integrar a IA de forma transversal em todas as etapas do SDLC. Os alunos trocam as especificações e fazem uma avaliação de Q&A de outra Squad. O desenvolvimento iterativo envolve a colaboração humano-IA para: Codificação: pair programming com a máquina para autocompletar e gerar blocos de código. Plano de Testes e Depuração: geração de scripts de teste, análise de logs e identificação de padrões de erro, sempre com validação humana. Documentação e No-Code: geração automática de documentação técnica, APIs e exploração de plataformas low-code/no-code (como Lovable ou Claude Code). O módulo se encerra com uma revisão formal do projeto a partir das devolutivas do Q&A.

–Sprint 4 - Ética, Implantação e Futuro (Semanas 14-17): O foco se volta para as responsabilidades inerentes ao uso de IA e para a reta final do desenvolvimento. Os estudantes debatem criticamente questões de ética e governança, como vieses nos modelos, propriedade intelectual, plágio e privacidade de dados. Aborda-se a segurança no desenvolvimento (DevSecOps), ensinando como mitigar vulnerabilidades específicas de sistemas com IA, como o prompt injection. Além disso, os alunos preparam a implantação do sistema (estratégias de release e rollback) e realizam o desenvolvimento e refinamento final do projeto em equipe.

Após as quatro iterações, os alunos são orientados a estruturar o pitch e a demonstração do sistema para o P.O. (Semana 18), destacando como o uso da IA Generativa ajudou a resolver os desafios do projeto. A Fig 1. apresenta a síntese do processo.

Os critérios de avaliação do projeto foram estruturados em três níveis: a composição global da nota da disciplina, o cronograma de entregas divididas em dois pacotes, a saber:

Sprints 1 e 2 - Documento de Especificação do sistema (DOC1) com peso de 70%; - Entrega da 1ª versão parcial e funcional do sistema (Pacote 1) com peso de 30%.

Sprints 3 e 4 - Preparação e execução do plano de testes (DOC2) com peso de 20%; - Entrega da 2ª versão parcial e funcional do sistema (Pacote 2) com peso de 30%; - Entrega da versão final do sistema (Pacote 3) com peso de 50%.

A avaliação do Documento de Especificação foi detalhada e dividida em 7 critérios de qualidade e engenharia:

–Organização e Estrutura do Documento: Exige sumário atualizado, glossário de termos para evitar ambiguidades, histórico de revisões (controle de versão) e obediência a um padrão institucional ou de mercado;

–Compleitude e Escopo: Requer a definição clara dos objetivos do sistema, fronteiras (o que o sistema não fará), identificação de todos os perfis de stakeholders e mapeamento de pendências com responsáveis e prazos;

–Requisitos Funcionais e Cobertura: Avalia a descrição dos cenários de sucesso e exceção, a consistência e isolamento das regras de negócio, a modelagem inicial de dados e o fluxo de trabalho para processos complexos;

–Rastreabilidade de Artefatos: Exige identificação única para cada requisito, uso de uma Matriz de Rastreabilidade (RTM) ligando requisitos a objetivos de negócio e o mapeamento de dependências entre os requisitos;

–Requisitos Não Funcionais: Avalia métricas de desempenho (tempo de resposta/latência), segurança (níveis de acesso e criptografia), escalabilidade (SLA e crescimento) e usabilidade (protótipos e acessibilidade);

–Viabilidade e Riscos Técnicos: Considera restrições tecnológicas, documentação de integrações com APIs e terceiros, além de suposições e dependências externas que podem impedir o desenvolvimento;

–Artefatos Solicitados: Verifica a notação, conformidade e cobertura semântica de artefatos essenciais, como Diagramas de Caso de Uso e sua especificação textual, Wireframes, Diagramas de Sequência e Diagrama de Domínio.

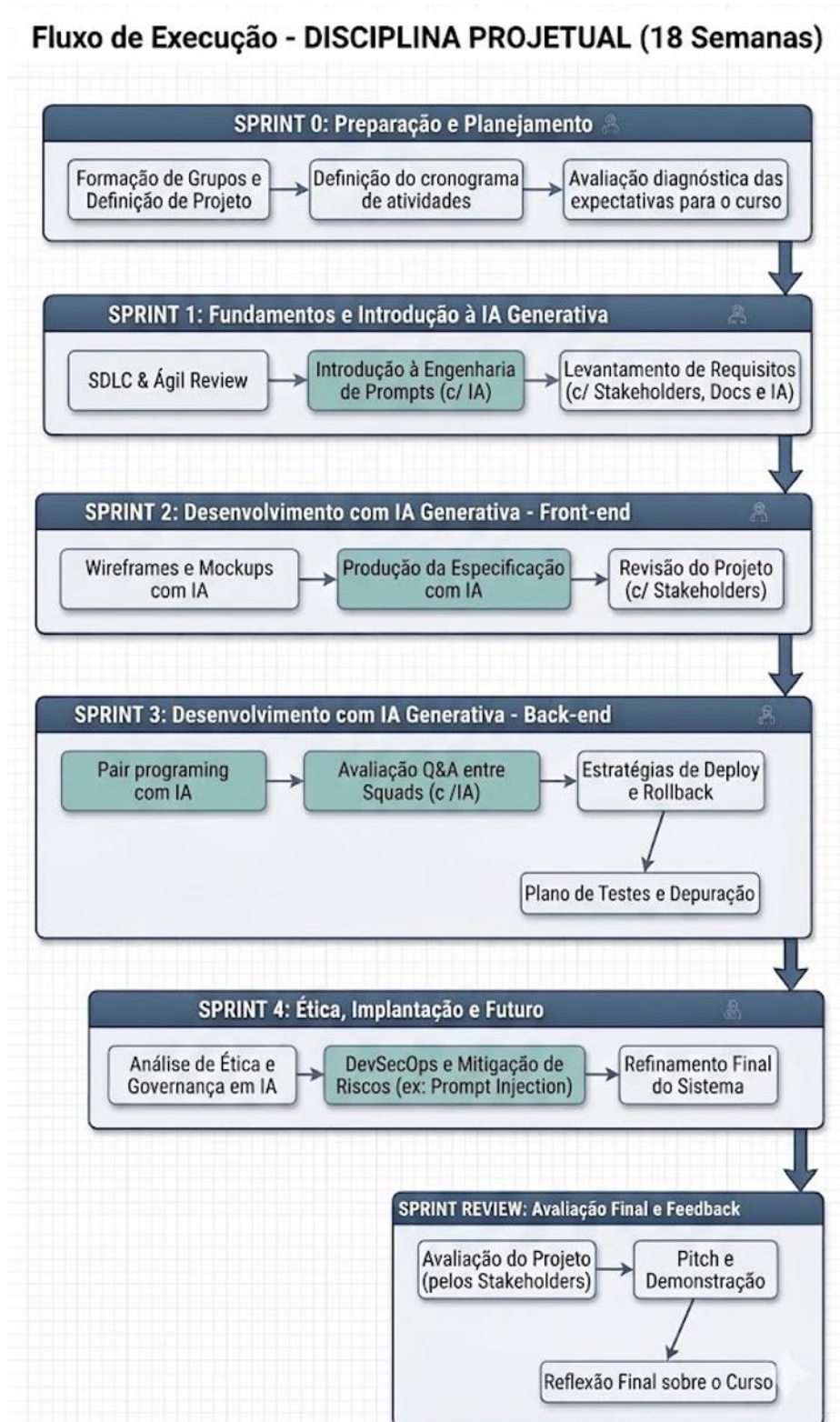


Fig 1. Síntese das iterações do projeto. Elaborado pelo autor

De forma transversal, um fator considerado em todas as entregas foi que, por se tratar de uma disciplina experimental, exige-se que as tarefas desenvolvidas com o apoio da IA generativa tenham seus prompts registrados e que os resultados passem por avaliação e discussão crítica. Outro mecanismo de avaliação ocorre na transição de entregas, por meio de sessões de Peer Review (Q&A entre os Squads) para avaliar os relatórios de garantia de qualidade e garantir que a dívida técnica gerada seja devidamente tratada.

A disciplina é encerrada com um debate reflexivo sobre os aprendizados, os desafios enfrentados, e o impacto futuro da IA nas perspectivas de carreira e na sociedade.

#### 4. Discussão

Ao ensinar Engenharia de Software promovemos a transição do aluno para um patamar profissional, mais alinhado com o perfil que o mercado procura. Quando agregamos IA ao pacote de ensino, estamos de certa forma aproximando os hard skills da formação acadêmica aos hard skills desejados por contratantes com o uso destas ferramentas.

Isso sempre foi feito no âmbito das disciplinas práticas. Ferramentas Case (Computer-Aided Software Engineering) foram agregadas ao ensino de Engenharia de Software logo que surgiram. Fato semelhante ocorreu com as ferramentas geradoras de código desde as primeiras versões. Isso sem deixar de ensinar como fazer sem apoio destas ferramentas.

Na entrevista com o pesquisador Martin Pinzger (Universidade de Klagenfurt), ele aponta que a transição de um desenvolvedor júnior para sênior é um reflexo direto do que acontece na educação universitária. A IA é essencialmente um vasto modelo probabilístico, o que significa que sempre existe um grau de incerteza. Essa incerteza pode levar a falhas em sistemas de software, e simplesmente não podemos nos dar ao luxo disso — principalmente em sistemas críticos para os negócios que precisam operar corretamente [Pinzger, 2026].

Como a IA agora resolve essas tarefas simples instantaneamente, os alunos correm o risco de desenvolver uma "dependência cognitiva" da máquina. Isso gera um sério hiato no mercado: a indústria precisará cada vez mais de desenvolvedores seniores para auditar e validar o código gerado por IA, mas as bases de treinamento dos juniores estão sendo terceirizadas para a automação, ameaçando a formação de novos talentos.

Observamos que a prática de conceitos anteriormente apresentados é importante, como um processo em espiral que repetimos várias vezes, cada vez com mais eficácia, acumulando experiências práticas que produzem a sinapse desejada para os conceitos previamente explorados.

No contexto da disciplina projetual relatada, observamos melhoria dos produtos apresentados no final do semestre letivo. As ferramentas de IA adotadas promovem, de fato, notável melhoria para os produtos documentais e os artefatos de software produzidos.

Mas o objetivo é o aprendizado. Neste aspecto, observamos a cada aula, o crescimento do interesse sobre o tema. A cada entrega, os artefatos gerados são produzidos como mais riqueza de detalhes do que a experiência dos alunos pode produzir. Este

descompasso, abre espaço para uma discussão mais ampla sobre os resultados alcançados.

Os alunos foram estimulados a questionar a IA sobre detalhes que eles não pensaram em incluir no projeto. Em um relato, o aluno questionou a IA sobre um requisito foi considerado ser entregue por um sistema externo. A IA respondeu que, partindo da premissa que "ela" era um arquiteto sênior (isso foi colocado no Prompt) esse posicionamento era esperado para este papel.

Outro ponto interessante foi pedir para uma Squad avaliar o projeto da outra usando IA. Isso abriu várias discussões sobre o entendimento das necessidades, porque, ainda que as Squads tinham acesso os mesmos recursos computacionais e documentos, havia muita divergência de entendimento. Ou seja, a IA não resolveu, questões importantes que um arquiteto experiente poderia resolver.

É evidente que a construção dos Prompts também impactaram nos resultados. Os grupos mais dedicados nesta tarefa tiveram resultados melhores.

Na oferta do segundo semestre, os professores também estavam mais experientes para as cobranças, discussões, observações sobre prompts e condução do projeto. Os resultados da segunda turma foram mais próximos do esperado e mais completos.

Sobre as entregas do grupo do primeiro semestre, houve muita preocupação com a documentação e os alunos acabaram por minorizar o funcionamento da aplicação implantada em nuvem. Eles negligenciaram alguns requisitos não funcionais solicitados, como o uso do "Java Spring boot" e o "Keycloak" para o gerenciamento de identidade. Optaram por seguir com "Typescript" no Backend porque as ferramentas adotadas tinham dificuldade para gerar o Backend nos padrões definidos.

Observou-se uma espécie de combinação entre os grupos, na esperança de que o professor não fosse penalizar as escolhas arquiteturas. Três grupos apresentaram a aplicação em notebooks pessoais, rodando em máquina local. Os resultados da aplicação atendiam 90% dos requisitos funcionais identificados. Estas questões foram discutidas em detalhes na Sprint Review e os grupos foram penalizados com redução de nota.

Na segunda turma, os professores foram mais incisivos na cobrança dos requisitos não funcionais e os trabalhos andaram melhor. Nesta rodada, todos os grupos finalizaram as aplicações rodando em uma nuvem "AWS". Dois dos quatro grupo foram até o final com o "Keycloak" e um grupo utilizou uma aplicação Open Source para resolver a validação de usuários com fator duplo. Um dos grupos, embora tivesse concluído o desenvolvimento, apresentaram a arquitetura informando que utilizaram o "Supabase" (algumas IAs sugerem este banco de dados open source como opção), quando haviam duas opções para armazenamento: "PostgreSQL" ou "MariaDB". Por fim, o grupo teve muita dificuldade para apresentar o produto. Eles não se prepararam para o pitch e foram também penalizados neste quesito. Os demais grupos foram bem, com pequenos pontos de melhoria identificados nos produtos apresentados.

Outro ponto importante que cabe nesta discussão. Em versões anteriores da oferta desta disciplina, os alunos não finalizavam os projetos. Havia uma entrega parcial devido ao tempo. Com a intervenção da IA, observamos mais completude nas entregas. O tempo de 18 semanas foi mais confortável para os grupos dado que o tempo de elaboração dos

artefatos diminuiu e a produtividade aumentou. Isso estimulou os alunos para a finalização dos projetos.

Nesta segunda turma, um aluno do curso de mestrado, desenvolvedor sênior, com 20 anos de experiência em desenvolvimento de aplicações foi estimulado a fazer o mesmo projeto, nas mesmas condições que os alunos da graduação. No dia do pitch ele apresentou a sua solução. Do ponto de vista de telas e navegabilidade, as soluções ficaram muito parecidas. As diferenças ficaram mais evidentes na composição dos requisitos. O aluno do mestrado, considerou pontos adicionais, que julgou interessante para apresentar aos stakeholders. Itens que não estavam descritos nos documentos, mas a experiência dele julgou interessante incluir.

Por exemplo, a aplicação foi desenvolvida para o contexto de uma Faculdade, mas ele entendeu que poderia incorporar flexibilidade para pequenas diferenças existentes em outras unidades acadêmicas, desta forma, ele ganhou a possibilidade de ampliar o uso da aplicação. Os alunos da graduação, não consideraram este tipo de inclusão, atenderam estritamente o que estava escrito.

A discussão de finalização ficou mais rica quando o aluno do mestrado mostrou um relatório de entrega do produto gerado pela IA. O relato indicava que a produção da aplicação teve a duração de 20 horas apenas para a finalização do produto e fazia uma comparação com um desenvolvedor sênior, que gastaria de 40 a 60 dias úteis para produzir o mesmo aplicativo sem uso de IA.

Ele também relatou ter utilizado 43 milhões de tokens para gerar a aplicação completa. Ele não escreveu nenhuma linha de código. Este ponto é um limitante para o aluno de graduação que não possui uma conta de IA paga.

## 5. Conclusões

Este trabalho relatou experimentos empíricos sobre o uso de IA em disciplinas projetuais de Engenharia de Software.

Neste escopo foram observados os impactos da integração da IA Generativa no aprendizado prático de projetos de software por meio da execução das etapas do SDLC com a produção de artefatos gerados por IA e análise crítica dos artefatos gerados durante o processo proposto.

Os resultados em um ano de experimentos são interessantes posto que: 1) os alunos cumpriram as demandas solicitadas; 2) a análise crítica dos resultados foi promovida pro meio de debates a cada aula; 3) Os alunos revisitaram todas as etapas do SDLC; 4) As ferramentas de IA foram utilizadas de modo satisfatório, conforme orientações dos professores.

A integração dessas tecnologias no ensino formal de Engenharia de Software ainda carece de maturidade metodológica. Na maioria das vezes, a IA é vista pelos estudantes apenas como um "autocompletar" avançado ou um atalho para a codificação, desvinculada do rigor metodológico da engenharia. O distanciamento entre a teoria clássica do SDLC – que exige fronteiras de sistema bem definidas, regras de negócio consistentes e avaliação de riscos técnicos – e o uso puramente instrumental da IA cria um gap perigoso.

Há uma deficiência no ensino sobre como estabelecer uma verdadeira colaboração humano-IA, onde o pensamento crítico seja o alicerce para a validação rigorosa de artefatos e códigos gerados por IA, mitigando riscos de segurança (como prompt injection ou vulnerabilidades em dependências) e dilemas éticos (vieses e propriedade intelectual).

A revisão do processo de software ganhou novas ferramentas de modo a capacitar futuros profissionais não apenas a utilizar a IA para acelerar o desenvolvimento (incluindo abordagens low-code/no-code), mas para atuar como revisores críticos e arquitetos de soluções seguras e éticas em um novo ecossistema tecnológico.

A título de trabalhos futuros pretende-se construir indicadores para acompanhar este processo de modo mais acurado.

### **Agradecimentos**

Este estudo foi desenvolvido no âmbito das pesquisas do programa de pós-graduação em Computação Aplicada da Universidade Presbiteriana Mackenzie.

### **Declaração sobre IA generativa e tecnologias assistidas por IA no processo de preparação do manuscrito.**

Durante a preparação deste trabalho, os autores utilizaram Google NotebookLm e Google Gemini para brainstorming de estrutura do artigo, revisão da redação e melhoria da fluidez do texto. Após a utilização desta ferramenta/serviço, o autor revisou e editou o conteúdo conforme necessário e assume total responsabilidade pelo conteúdo do artigo publicado.

### **Referências**

- Goel, Pawan Kumar and Komal, Km and Vashishth, Nitish (2024). AI-Driven Software Development Lifecycle Optimization. In *Advances in Systems Analysis, Software Engineering, and High-Performance Computing*, pages 70–86. IGI Global.
- Boehm, B. (2006). A view of 20th and 21st century software engineering. In *Proceedings of the 28th international conference on Software engineering*, pages 12–29, Shanghai China. ACM.
- Bordin, A. S. (2024). Ensino de Processo de Software em Colaboração com Empresas: Uma Experiência Potencialmente Extensionista. In *Anais do IV Simpósio Brasileiro de Educação em Computação (EDUCOMP 2024)*, pages 153– 160, Brasil. Sociedade Brasileira de Computação.
- Córdova, P. R., Baade, J. H., and Santos, A. M. D. (2020). Problem-based learning (PBL) e interdisciplinaridade no ensino-aprendizagem de engenharia de software. *Revista Ibero-Americana de Estudos em Educação*, 15(4):2078– 2093.
- Fernandez, D. L., Raya, L., Ortega, F., and Garcia, J. J. (2019). Project based learning meets service learning on software development education. *The International journal of engineering education*, 35:1436–1445.

Kozub, V., Druzhynin, V., Trufanova, D., Ihnatenko, P., and Kolos, K. (2025). Using artificial intelligence in software development processes: achievements and challenges. *Sustainable Engineering and Innovation*, 7(2):463–476.

Lopes, F. S., Eliseo, M. A. (2022). Software Residency Practices as a Complement to the Teaching-Learning Process in Software Engineering: An Experience Report. 2022 17th Latin American Conference on Learning Technologies, LACLO 2022.

Miao, F., Cukurova, M. (2024). AI competency framework for teachers. UNESCO.

Pinzger, M. (2026). Artificial Intelligence in Software Development: “AI Is Swimming in Its Own Soup.”.

Ramos Da Silva, K., Sergio De Oliveira Barbosa, L., Lira Botelho, W., Mateus Barbosa Pinheiro, J., Dos Santos Peixoto, I., and Vitoria Coimbra Borges De Menezes, I. (2023). INTELIGÊNCIA ARTIFICIAL E SEUS IMPACTOS NA EDUCAÇÃO: UMA REVISÃO SISTEMÁTICA. *RECIMA21 - Revista Científica Multidisciplinar* - ISSN 2675-6218, 4(11):e4114353.

Varna, Armyanova, M., Aleksandrova, Y., University of Economics - Varna, Varna, Bulgaria (2023). Using Artificial Intelligence in Software Development. *Izvestia Journal of the Union of Scientists - Varna. Economic Sciences Series*, 12(1):167–176.