Cuaderno Activa
Revista científica de la Facultad de Ingeniería

Uso da Ferramenta PlantUML no ensino de UML em Engenharia de Software: um estudo sobre adoção de ferramentas de modelagem baseadas em script

Use of the PlantUML Tool in Teaching UML in Software Engineering: a study on the adoption of script-based modeling tools

Luiz Carlos Machi Lozano[1], Fabio Kawaoka Takase [2], Fabio Silva Lopes [3]

Tipo de Artículo: Investigación revisión.

Recibido: 00/00/0000. Aprobado: 00/00/0000. Publicado: 00/00/0000

Resumo: O aprendizado da UML (Unified Modeling Language) para modelar e documentar uma solução de software, sem ambiguidade e de modo colaborativo entre estudantes consiste em habilidade importante para uma boa formação em Engenharia de Software. No escopo deste estudo, desenvolvemos de modo empírico uma abordagem baseada no uso da UML por meio da implementação baseada em scripts no formato PlantUML como base para modelagem e documentação de projetos com foco no aprendizado de Engenharia de Software para alunos da graduação. No período de um ano, a abordagem foi testada em 4 ambientes distintos que envolvem alunos de graduação de diferentes etapas e cursos da área de Computação. Entre os resultados observados, estão registradas a baixa dificuldade de adoção da ferramenta, a melhora de qualidade dos diagramas UML produzidos e o entendimento mais efetivo sobre integração de ferramentas no âmbito do processo de desenvolvimento de produtos de software.

Palavras-chave: Aprendizagem de modelação; Engenharia de software; UML baseado em texto legível por humanos.

Abstract. Learning UML (Unified Modeling Language) to model and document a software solution—without ambiguity and in a collaborative way among students—is an important skill for solid training in Software Engineering. In the scope of this study, we developed an empirical approach based on the use of UML through script-based implementation using the PlantUML format as a foundation for project modeling and documentation, focused on Software Engineering learning for undergraduate students. Over the course of one year, the approach was tested in four distinct environments involving undergraduate students at different stages and from various Computing-related courses. Among the observed results were low difficulty in adopting the tool, improved quality of the produced UML diagrams and more effective understanding of tool integration within the software product development process.

Keywords: Modeling Learning; Software Engineering; human-readable text-based UML.

[1] Autor correspondiente: Luiz Carlos Machi Lozano. Filiación institucional: Universidade Presbiteriana Mackenzie. País: Brasil, Ciudad: São Paulo. Correo electrónico: luiz.lozano@mackenzie.br ORCID: 0000−0003−0464−3292

[2] Autor correspondiente: Fabio Kawaoka Takase. Filiación institucional: Universidade Presbiteriana Mackenzie. País: Brasil, Ciudad: São Paulo. Correo electrónico: fabio.takase@mackenzie.br ORCID: 0009−0006−4187−796X

[3] Autor correspondiente: Fabio Silva Lopes. Universidade Presbiteriana Mackenzie. País: Brasil, Ciudad: São Paulo. Correo electrónico: fabio.lopes@mackenzie.br ORCID: 0000−0001−8274−7682

I. Introdução

A prática de Engenharia de Software envolve o trabalho colaborativo de profissionais com diferentes competências durante todo o ciclo de vida de uma solução de software. Independentemente das definições de fases do ciclo de vida definidos por necessidades de controle e governança de uma organização e das metodologias adotadas para a construção do software, a comunicação assertiva e sem ambiguidade entre os diversos participantes do projeto sobre o objeto em construção é essencial.

Dentre os diferentes níveis de formalismos aplicáveis à descrição de soluções de software, a Unified Modeling Language (UML) é bastante popular e permite representar diferentes visões do software em construção, incluindo aspectos estruturais e dinâmicos de especificação, projeto e implantação de software. Esta popularidade se traduz em disponibilidade de diversas ferramentas para construção das visões da UML, tornando-a um bom recurso para modelagem de produtos, bem como, para comunicação entre os stakeholders do projeto.

A área de Engenharia de Software é relativamente nova quando comparada a outras como matemática ou Química. Desde o seu formalismo como área de estudo, no final dos anos 1960, há um esforço contínuo na consolidação de conceitos que vão além daquilo que o termo "engenharia" preconiza, passando pelo estudo da complexidade do software, aspectos práticos da produção de software, ferramentas, plataformas, padrões e modelos abstratos.

Não obstante, o ensino nesta área é desafiador dada a natureza dinâmica e complexa envolvida. A formação básica em Engenharia de Software deve contemplar a apropriação de conceitos, habilidades práticas e aprimoramento de softskills como liderança, trabalho em equipe e empatia [13].

Outro ponto está no custo dos produtos. Na medida que as ferramentas ganham projeção de mercado elas também evoluem o modelo de negócio e relação com o mercado. Iniciam como uma ferramenta aberta, mantida por uma comunidade, depois mudam para uma modalidade Freemium, com algumas features gratuitas e outras pagas. Por fim, a ferramenta se torna 100% paga. Também há exemplos de oferta para uma versão acadêmica com custo reduzido.

Ocorre que o custo para universidade aumenta e, nem sempre a ferramenta escolhida é aquela que o estudante vai encontrar no estágio ou novo emprego. Portanto, o desafio está no equilíbrio financeiro do processo de ensino para a universidade, considerando a melhor prática agnóstica, sem a dependências de uma ferramenta específica.

De modo complementar, existem questões de acessibilidade envolvidas. Estudantes com limitações visuais tem dificuldade em ler diagramas, mas conseguem ter acesso mais fácil a linguagem de script, como é utilizada pelo PlantUML [11].

Vale salientar que o aprendizado de UML para especificação e projeto de soluções de software é um desafio porque implica em trabalhar com duas aprendizagens distintas: como especificar o projeto e como utilizar a UML corretamente. O uso do PlantUML aplicado em projetos de disciplinas de Engenharia de Software na graduação pode ser um diferencial para melhorar as habilidades profissionais neste tema. Contudo, há carência de trabalhos que apontem nesta direção, abrindo oportunidades de pesquisa neste contexto.

Logo, a questão de pesquisa levantada neste estudo foi a seguinte: A utilização de linguagem de script com independência de ferramentas de mercado no ensino de modelagem de produtos de software pode contribuir para melhores resultados de aprendizagem?

Considerando os pontos apresentados, este estudo objetivou experimentar o produto PlantUML em ambientes de ensino distintos para avaliar resultados de aprendizagem em disciplinas que utilizam UML para apoio na especificação de soluções de software.

Experimentalmente, utilizamos o PlantUML por 6 meses com estagiários do laboratório de pesquisa na área de Engenharia de Software. Neste Ambiente 1, implantamos o GitLab na nuvem onPremise da Universidade e elaboramos um guia de referência para uso da ferramenta no laboratório, que foi vinculado à capacitação de onboarding para novos participantes dos projetos que executamos no laboratório. Em seguida iniciamos os processos de modelagem e documentação dos produtos já existentes do laboratório. Na etapa seguinte, implantamos esta abordagem para o programa de Residência de Software que o Laboratório oferece para estudantes da graduação, onde são desenvolvidas aplicações para clientes reais. Neste programa temos dois grupos atuando em paralelo. Residência de Software presencial e Residência de Software on-line para alunos dos cursos EaD (Ambientes 2 e 3, respectivamente). Por fim, a abordagem foi implementada em disciplinas da graduação, nos cursos de Sistemas de Informação, Ciência da Computação e Análise e Desenvolvimento de Sistemas (Ambiente 4).

Este artigo foi estruturado em cinco seções principais. Inicialmente, a Seção 1 contextualiza a pesquisa, delineando o problema investigado e os objetivos do estudo. A Seção 2 apresenta a revisão da literatura, sintetizando os conceitos-chave e os estudos correlatos relevantes para a pesquisa. A Seção 3 detalha a metodologia empregada, incluindo o delineamento do estudo e os procedimentos de análise e composição de

resultados. A Seção 4 apresenta os resultados obtidos e, por fim, a Seção 5 discute as conclusões do estudo, suas implicações e as direções para pesquisas futuras.

II. Sistemas de suporte para o ciclo de vida de um produto de software

É consenso que modelar software constitui uma boa prática. Muitos pesquisadores que atuam como engenheiros de software já publicaram artigos evidenciando o uso de diagramas UML no desenvolvimento de sistemas e as respectivas contribuições, a fim de promover o avanço da Engenharia de Software como área de conhecimento. O estudo de Koç e colegas [6], abordou a questão em uma revisão sistemática da literatura na área de pesquisa sobre Engenharia de Software que buscou entendimento sobre quais diagramas UML são populares, por que são utilizados e quais áreas de aplicação são as mais populares. Em 247 publicações ao longo de 20 anos (entre 2000 e 2019) apresentavam diagramas UML. Em 68,7% delas a UML foi utilizada para modelagem de aplicações.

Desde 2000, muito se avançou após a padronização da modelagem e comunicação proposta pela UML 1.1 [9]. No que diz respeito ao ensino de modelagem isso não foi diferente. A prática da modelagem já era vivenciada em cursos de computação desde a década de 1980, por meio da Análise Estruturada, com diagramas do tipo DFD (Chris Gane ou Tom DeMarco), Modelos Entidade-Relacionamento (Peter Chen) e diagramas de decomposição (Meillier Page-Jones). Aos poucos, estes diagramas foram perdendo espaço para os diagramas da UML pois o mercado já lidava com a orientação a objeto e a prática estruturada perdia força.

Por outro lado, as ferramentas de CASE (Computer-Aided Software Engineering) começaram a proliferar e chegaram também às aulas de laboratório de Engenharia de Software. Algumas eram somente diagramáticas e outras permitiam interação com o código. Alguns exemplos de ferramentas utilizadas estão listados na sequencia.

- StarUML.
- WhiteStarUML.
- Enterprise Architect.
- Lucidchart.
- Draw.io.
- Miro.
- UMLet.

Com várias opções disponíveis, o estudante tem liberdade para escolher qual ferramenta utilizar. Porém algumas opções apresentam limitações na implementação correta da simbologia definida pela OMG. Isso dificulta o processo de ensino e a correção de atividades. O trabalho de Cunha. et al. [2] discute parcialmente esta questão. No estudo, eles avaliaram 5 ferramentas e observaram que as ferramentas que estão mais alinhadas com as diretrizes da norma, geram menor quantidade de erro no processo de modelagem.

A abordagem ágil aproximou a documentação de projeto e o desenvolvedor, levando diagramas e descrições para o dia a dia do desenvolvedor [3]. Estes diagramas, neste contexto assumem um papel mais assertivo na comunicação e descrição de decisões tomadas em consenso, em contraste com uma abordagem orientada a planos em que muita documentação é gerada sem o foco na comunicação com desenvolvedores e sem a participação de desenvolvedores nas decisões tomadas. Os diagramas gerados neste novo contexto com foco na comunicação e colaboração fazem portanto parte do produto em desenvolvimento, devendo ser versionado e rastreado em seu ciclo de vida junto com outros artefatos do projeto, como por exemplo o código fonte e bibliotecas em uso.

No ambiente de desenvolvimento profissional de software, a colaboração e a comunicação clara e sem ambiguidade é essencial, e em um contexto em que a linguagem de modelagem é unificada, no dia a dia a utilização de uma ferramenta única para descrição e projeto de uma solução de software é importante. Este suporte de ferramenta, linguagem e consenso entre diferentes participantes do projeto deve existir durante todo o ciclo de vida do produto de software. Sem este suporte operacional, o reuso sistemático de artefatos de projetos já construídos por diferentes equipes/squads se torna uma tarefa complexa [12].

2.1 Sobre o PlantUML

O PlantUML é uma ferramenta Open Source cuja finalidade é renderizar diagramas a partir de uma linguagem de texto. Foi desenvolvida como Open Source em 1996 por Arnaud e mantém um repositório público no GitHub desde 2010 onde a ferramenta está disponível e apresenta as categorias de licenciamento, entre elas, GPL (General Public License), Apache e MIT [11].

O diferencial desta ferramenta está justamente na possibilidade de criar diagramas a partir de uma gramática formal projetada para análise, validação e integração com outras ferramentas, entre elas, GitHub, GitLab e VS-Code. Isso permite renderizar os diagramas nestas ferramentas de modo a manter código e documentação no mesmo repositório.

De modo complementar, o PlantUML dá suporte para outros diagramas não inclusos na UML, como por exemplo Gráficos de Gantt, Archimate (TOGAF), Mind Maps e WBS (Work Breakdown Structure).

2.2 Exemplos de aplicações relacionadas com PlantUML

Alguns estudos seguem na possibilidade de utilizar IA para converter imagens criadas manualmente em artefatos digitais passiveis de leitura a compreensão. O estudo de Conrady e Cabot [1] aprofunda o tema por meio de experimentos de submissão de imagens a um LLM multimodal para interpretar e gerar o script em PlantUML.

Já os estudos de Fill. et al. [5] segue a linha da geração de diagramas a partir de prompts executados em ferramentas como o ChatGPT, que são capazes de gerar um script em PlantUML. Na visão dos autores, os assistentes de IA estão se tornando uma nova interface para interagir com a modelagem, em linguagem natural.

Uma outra abordagem foi evidenciada no estudo de Romeo e colegas [10]. Eles desenvolveram uma ferramenta para compreender desvios de Design e Implementação e Documentação (DID), conectando referências UML descritas em PlantUML às entidades de código-fonte correspondentes (por exemplo, classes Java), a ferramenta se baseia em novas métricas de cobertura entre os artefatos UML e o sistema.

Observa-se que os exemplos supracitados são úteis para o contexto ensino-aprendizagem pois podem ser utilizados em abordagens práticas de projetos de soluções modeladas em UML, onde o uso de formatos UML baseados em texto são considerados.

III. Método

Neste trabalho relatamos a experiência de aplicação do PlantUML em 4 grupos diferentes de alunos.

O uso da UML nas atividades de projeto foram realizadas seguindo as 4+1 visões propostas por Kruchten [7].

Em cada grupo de alunos um tema único de projeto foi proposto e traba-lhado de forma colaborativa, enfrentando diferentes desafios de aprendizagem, comunicação e interação interpessoal.

As ferramentas utilizadas nestas ações foram o GitLab para trabalho colabo-rativo com controle de versão para os artefatos de projeto, o PlantUML para a renderização dos scripts escritos pelos alunos para apresentação dos diagramas UML e o MS TEAMS para comunicações e documentos auxiliares.

No trabalho realizado no Ambiente 1 (Estagiários do laboratório), três esta-giários trabalharam na implantação de uma instância do GitLab com integração com o PlantUML e na sustentação da operação de todo ambiente de suporte, incluindo gerenciamento de acessos e usuários, backup diário e monitoramento. O grupo de estagiários e os dois laboratoristas também utilizam estas ferramentas para sustentar as atividades de desenvolvimento e manutenção de operacionalidade de outras ferramentas e soluções implementadas e sustentadas pelo laboratório.

Os Ambientes 2 e 3 são ambientes relacionados ao programa de Residência de Software do laborário [8] que aplica o PBL (Problem Based Learning) à Engenharia de Software [4].

A dinâmica de trabalho foi suportada por estagiários que organizavam reuniões diárias, e por reuniões semanais de acompanhamento com o coordenador e professores. Estas reuniões semanais no Ambiente 2 foram realizadas presencialmente e no Ambiente 3 remotamente através do MS Teams. O GitLab serviu como repositório central para todos os artefatos e código-fonte.

Para a coleta de dados, foram registrados os seguintes indicadores: (1) a taxa de evasão do grupo ao final do semestre e (2) uma análise qualitativa e comparativa da qualidade dos artefatos de modelagem produzidos em relação aos das turmas de outros ambientes.

A Residência presencial (Ambiente 2) contou com um grupo de 18 alunos em seu início que assumiu o desenvolvimento do projeto ’Interprèt’, seguindo um ciclo iterativo guiado pelo modelo proposto por Kruchten, com um trabalho inicialmente focado na Visão de Cenários, com trabalho posterior sobre a Visão Lógica e de Processos para no final, construir as visões de Desenvolvimento e Implantação durante os Sprints de implementação. O rito de reuniões diárias foi acompanhado pelos estagiários do laboratório, que cumpriram o papel de Scrum Master nas squads. Semanalmente uma reunião presencial com todos os participantes das squads e os professores do laboratório foi realizada.

No Ambiente 3 (Residência on-line) um grupo de nove alunos iniciou o desenvolvimento do projeto ’Gerenciamento de Competências’, seguindo um ciclo iterativo guiado pelo modelo arquitetural 4+1 de Kruchten [7]. A primeira fase focou na Visão de Cenários, com levantamento de requisitos e criação de diagramas de casos de uso e protótipos. Em seguida, a Visão Lógica foi detalhada com diagramas de classe de domínio e de sequência, todos gerados via PlantUML.

O ambiente 4 (Cursos de Graduação) envolveu a aplicação da abordagem em diversas turmas.

Na disciplina de Desenvolvimento de Sistemas do curso de Sistemas de Informação o experimento comparou uma turma de 2024 (grupo de controle) com uma de 2025 (grupo experimental) para avaliar o impacto da adoção do PlantUML. A disciplina, em ambos os anos, foi estruturada em dois bimestres, com a nota de cada bimestre sendo composta 50% por entregas de

um projeto e 50% por uma prova. O projeto consistia na modelagem de um sistema a partir de requisitos, sem a etapa de codificação, simulando um ambiente de engenharia de software. Para isolar a variável de estudo, o nível de dificuldade dos projetos e das provas foi mantido equivalente para ambas as turmas. A diferença central foi o ecossistema de trabalho: o grupo de controle utilizou a ferramenta visual StarUML com entregas no ambiente virtual de aprendizagem Moodle, enquanto o grupo experimental utilizou o PlantUML integrado ao GitLab, com suporte adicional de vídeos explicativos gravados pelo professor.

Na disciplina de Engenharia de Software do curso de Ciência da Computação, o ambiente foi utilizado para a realização do projeto da disciplina em três turmas distintas. Esta disciplina se encaixa entre uma disciplina que introduz o uso de diagramas UML (Casos de uso, sequência, classes, comunicação, atividade e estados) e outra disciplina que trabalha com a implementação da solução especificada. Esta disciplina tem como principal foco a apresentação aos alunos dos processos de desenvolvimento de software, metodologias, disciplinas envolvidas, ciclo de vida de um produto de software e qualidade de software. Para cada turma foi proposto um tema único de projeto para exigir um trabalho colaborativo entre diversas squads. O projeto consistia na modelagem de um sistema a partir do problema proposto como tema do projeto, desde identificação de stakeholders, caracterização do problema, identificação de cenários e restrições, elicitação de requisitos funcionais e não funcionais até a definição de uma arquitetura de software com uma visão de implantação, sem a codificação da solução. Nesta disciplina o GitLab serviu como repositório central para todos os artefatos e código-fonte e os diagramas UML foram gerados em arquivos tipo markdown versionados e renderizados no próprio GitLab utilizando o PlantUML. Para avaliar o impacto da realização das atividades de projeto no aprendizado do aluno foi coletada informação sobre a participação do aluno no projeto (intervenções no sistema do GitLab) e esta participação foi comparada à média das avaliações individuais.

IV. Resultados e Discussão

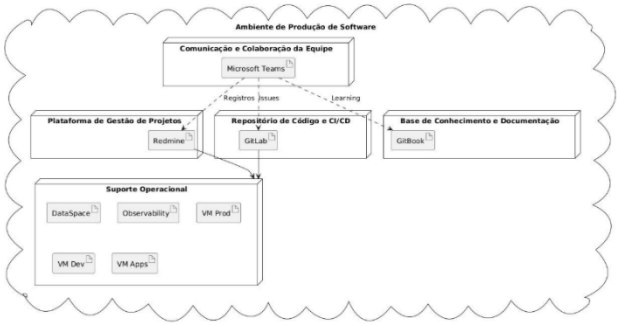
4.1. Ambiente 1 - Integrantes do Laboratório

Vale aqui relatar alguns dados do Laboratório de Estudos em Ambiente de Produção de Software da Universidade. Inicialmente o laboratório foi pensado como uma Fábrica de Software experimental. Desde 2010, alunos e professores vivenciam experiências reais de desenvolvimento de aplicações, praticando conceitos teóricos e práticos abordados em sala de aula, dos cursos de computação.

Em 2022 o Laboratório deixou de ser uma fábrica e adotou uma postura mais alinhada com a pesquisa com foco em Ambientes de produção de software. Para tanto, havia a necessidade de estruturar um processo que permitisse experienciar teoria e práticas em condições análogas ao mercado software.

Neste sentido, uma linha de produção foi planejada considerando ferramentas abertas e processo flexível, capaz de atender complexidades distintas de cada projeto novo. A Figura 1 apresenta os artefatos da arquitetura e suas relações.

Figura 1. Artefatos do Ambiente de Produção de Software



Observação: preparação própria.

Nesta configuração, o PlantUML é renderizado diretamente no GitLab. Portanto, os membros do laboratório mantém a documentação diagramática no mesmo repositório que o código, sob o mesmo controle de versões.

4.2 Ambiente 2 - Residência de Software presencial

O ambiente de residência de software é um ambiente de construção de software que mimetiza um ambiente de produção de software profissional e que deve portando adotar práticas de gestão de configuração e mudanças necessárias para estabelecer gestão e governança sobre soluções de software em construção e em operação.

Com relação à taxa de evasão, tivemos no período, uma evasão de 33%, iniciamos o semestre com 18 alunos e encerramos com 12. Esta taxa é elevada mas também esperada, pois como um projeto de extensão que exige uma dedicação semanal considerável, alunos que iniciam estágio em empresas durante o período da residência não conseguem conduzir as duas atividades em paralelo.

O uso do Git para versionamento, de *issues* e *kanban* para planejamento e controle de ações e documentação em *markdown* versionada e renderizada no próprio GitLab permitiu o estabelecimento rápido do rito diário de trabalho das *squads*, o que foi verificado por uma aceleração no processo se comparado à outras edições da residência. Em outras edições, o primeiro semestre foi dedicado exclusivamente ao entendimento do problema, levantamento de requisitos e especificação da solução. Nesta edição, além destas atividades, *sprints* de desenvolvimento foram realizados com avaliação de entrega parcial com *stakeholders* do projeto no primeiro semestre. Este resultado é impactante, pois nesta edição, foram definidas duas *squads* que trabalharam sobre o mesmo projeto e a comunicação e o trabalho colaborativo aconteceu de forma efetiva, como os resultados demonstram.

4.3 Ambiente 3 - Residência de Software *on-line*

Neste Ambiente, foi delineado como uma intervenção metodológica específica para os alunos dos cursos EaD. A abordagem foi criada em resposta direta ao problema de altas taxas de evasão (próximas a 100%) observadas em modelos anteriores que integravam alunos EaD com turmas presenciais.

O objetivo era criar um ecossistema de trabalho totalmente remoto para verificar o impacto no engajamento e na qualidade dos projetos. Um dos resultados mais expressivos observados no estudo proveio da primeira edição do programa de Residência *on-line*.

Historicamente, a tentativa de integrar alunos de EaD em turmas de residência presenciais resultava em uma taxa de evasão próxima a 100%, atribuída a dificuldades de engajamento em um modelo híbrido. A nova abordagem demonstrou um impacto significativo na retenção. O programa iniciou com 9 alunos e finalizou com 6, registrando uma taxa de evasão de aproximadamente 33%. Embora a evasão não tenha sido eliminada, sua drástica redução, quando comparada ao cenário anterior, representa um sucesso substancial na criação de um ambiente de engajamento viável para esses estudantes.

Adicionalmente, o grupo demonstrou alta performance, com uma análise comparativa indicando que a qualidade dos artefatos de modelagem e a profundidade no entendimento do problema foram superiores às das turmas híbridas dos anos anteriores. Como evidência dessa produtividade, a equipe remota conseguiu progredir até a etapa de implementação de um dos casos de uso ainda no primeiro semestre.

4.4 Ambiente 4 - Cursos de graduação

Desenvolvimento de Sistemas - Sistemas de Informação Foi conduzido um estudo de caso comparativo na disciplina de Desenvolvimento de Sistemas do curso de Sistemas de Informação. A disciplina é pedagogicamente estruturada para simular um ambiente real de desenvolvimento de software. O sistema de avaliação é dividido em dois bimestres e a nota de cada um deles é composta por duas partes de igual peso: 50% correspondem às entregas do projeto de modelagem e 50% a uma prova bimestral. O projeto semestral funciona como o pilar prático da disciplina. As equipes de alunos recebem a requisição de um novo software, incluindo uma descrição do problema e seus possíveis requisitos. A partir disso, eles devem realizar toda a modelagem do sistema, produzindo os artefatos UML necessários ao longo de seis entregas, que são distribuídas entre os dois bimestres. É importante ressaltar que o escopo da disciplina se concentra exclusivamente nas fases de análise e projeto, não envolvendo a implementação do código-fonte. Sobre essa estrutura fundamental, foi montado o experimento comparativo. Para garantir a validade da análise, ambas as turmas (2024 e 2025) trabalharam com projetos e provas de mesmo nível de dificuldade, definidos pelo professor. A principal variável investigada foi o ecossistema de ferramentas e a abordagem de trabalho. O grupo de controle (Turma 2024) utilizou a ferramenta visual StarUML com entregas via Moodle. O grupo experimental (Turma 2025), por sua vez, adotou o PlantUML integrado ao GitLab e recebeu suporte adicional através de vídeos explicativos sobre as ferramentas.

Tabela 1. Análise quantitativa do desempenho das turmas

Indicador	Turma 2024 (Controle)	Turma 2025 (Experimental)
Nº de Alunos	45	22
Média Final Geral (MF)	5,77	6,60

Observação: preparação própria.

Os dados da tabela 1 indicam uma melhora no indicador de desempenho da turma que utilizou a abordagem com PlantUML. Tendo em vista que o nível de dificuldade dos projetos e das avaliações foi controlado, a melhora observada na Média Final Geral, que saltou de 5,77 para 6,60 (um aumento de aproximadamente 14,38%), pode ser associada à mudança no ecossistema de ferramentas e no suporte pedagógico. A abordagem de "diagramas como código" do PlantUML, alinhada a práticas de desenvolvimento como o controle de versão no GitLab, pode ter contribuído para um maior engajamento e qualidade nas

entregas dos artefatos. Adicionalmente, o suporte assíncrono através de vídeos explicativos pode ter sido um fator importante para auxiliar os alunos a superarem as barreiras de aprendizado com a nova abordagem.

Engenharia de Software - Ciência da computação
Três turmas fizeram uso do PlantUML no projeto da disciplina de Engenharia de Software. Cada turma recebeu um único tema para desenvolver o projeto e as turmas foram divididas em *squads* de no mínimo 4 e no máximo 5 integrantes. Algumas exceções foram abertas formando grupos de 6 alunos para alunos que ingressaram tardiamente na disciplina. O número de participantes de cada turma e o número de *squads* podem ser visualizados na tabela 2.

Tabela 2. Turmas

Turma	Participantes	*Squads*
A	17	5
B	27	6
C	36	8

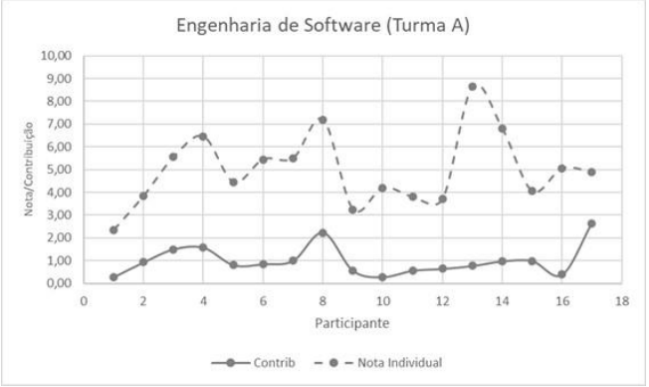
Observação: preparação própria.

Cada *squad* trabalhou sobre um ramo de trabalho (*branch*) separado no repositório do projeto da disciplina. Diferentes *sprints* foram configuradas de acordo com as 4 entregas planejadas para os projetos. O *backlog* de atividades das *sprints* foi criado e atualizado por cada *squads*, utilizando os recursos de planejamento disponibilizados pelo GitLab, como o uso de *issues*, *Kanban*, rótulos e *milestones*.

Nas Figuras 2, 3 e 4 a contribuição de cada participante e a nota individual de cada participante são plotadas para cada turma. A contribuição de cada participante foi calculada através do número de intervenções no projeto registradas no GitLab com uma ponderação sobre as intervenções sobre outros participantes do grupo (*squad)*. Um indicador de intervenções ponderadas bem próximo de 1 nestes gráficos indicam que os diversos membros do grupo de projeto atuaram de forma equilibrada, com um número de intervenções semelhantes. Foram consideradas as intervenções de *commit*, em que mudanças nos artefatos sob controle de versão são realizadas, e as intervenções de planejamento e acompanhamento de execução de tarefas, como a criação e atualização de *issues* no *Kanban* do projeto. A nota individual é a média das avaliações individuais de cada participante, sem a composição da nota final que considera as notas individuais e a nota de equipe do projeto.

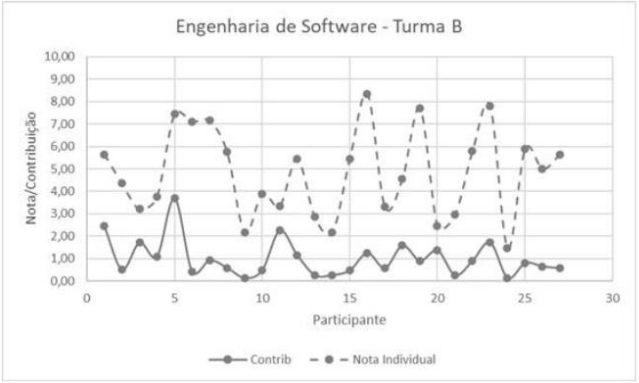
Através dos gráficos é possível verificar uma correlação entre o índice de colaboração registrado no projeto com as notas das avaliações individuais. Índices muito acima de 1 mostram um desequilíbrio na participação de membros de uma mesma *squad* e uma construção de artefatos que não foram utilizados como comunicação de conceitos e ideias entre os membros da equipe. Como podemos observar nos gráficos apresentados, índices próximos e acima de 1 estão relacionados com melhor desempenho nas avaliações individuais.

Figura 2. Contribuição e Nota individual dos participantes da turma A



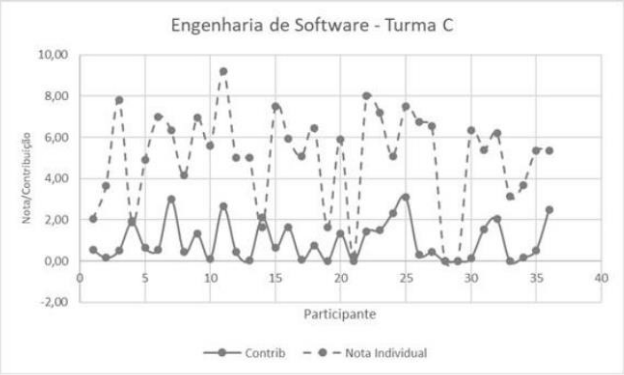
Observação: preparação própria.

Figura 3. Contribuição e Nota individual dos participantes da turma B



Observação: preparação própria.

Figura 4. Contribuição e Nota individual dos participantes da turma C



Observação: preparação própria.

Um exemplo do uso do PlantUML na construção de um diagrama de casos de uso é apresentado em 1.1.

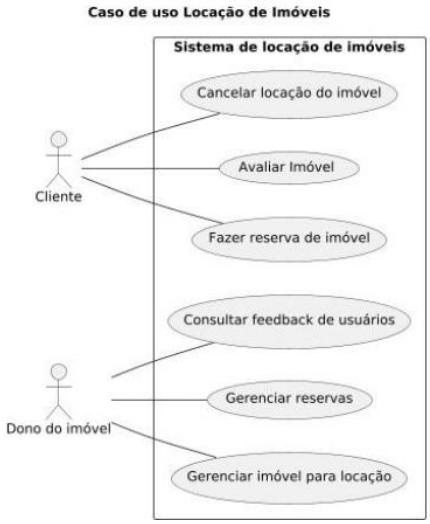
```
‘ ‘ ‘ plantuml @
startuml scale 0.8
left to right direction
title Caso de uso Locacao de Imoveis

Actor "Cliente" as us
Actor "Dono do imovel" as dn

rectangle "Sistema de locacao de imoveis" {usecase "Fazer reserv
        a de imovel" as UC1 usecase "Avaliar Imovel" as UC3
        usecase "Cancelar locacao do imovel" as UC4 usecase "Gerenciar
        imovel para locacao" as UC5 usecase "Gerenciar reservas" as UC7
        usecase "Consultar feedback de usuarios" as UC11 us -- UC1
        us -- UC3 us --
        UC4 dn -- UC5 dn
        -- UC7 dn -- UC11
}
@enduml ‘ ‘
‘
```

O diagrama de casos de uso renderizado é apresentado na Figura 5

Figura 5. Diagrama de casos de uso



Observação: preparação própria.

4.5 Benefícios observados

Um benefício prático e relevante observado com a adoção do PlantUML integrado ao GitLab foi a capacidade de gerenciar a documentação de software de forma análoga ao código-fonte.

Conforme o artigo já aponta como uma capacidade da abordagem, a ferramenta permitiu que os diagramas fossem efetivamente versionados e rastreados ao longo do ciclo de vida do projeto. Essa capacidade de rastrear, comparar e, se necessário, reverter alterações nos modelos foi um avanço significativo para a gestão da documentação. Adicionalmente, a equipe da Residência *on-line* explorou a criação de uma documentação rica utilizando arquivos no formato *Markdown* (.md).

Conforme a capacidade de integração da ferramenta, os alunos puderam incorporar os diagramas renderizados diretamente ao lado de textos explicativos e tabelas, prática que centralizou e facilitou a consulta e o entendimento do projeto, mantendo documentação e código no mesmo repositório.

Outro benefício do uso do PlantUML é indireto e foi observado nas três turmas de Engenharia de Software. Por ser uma ferramenta de *script* com versionamento integrado ao GitLab, foi possível monitorar a contribuição de cada participante no projeto, inclusive na elaboração de diagramas UML, o que permitiu correlacionar a nota individual de cada participante à sua contribuição no projeto.

O uso de uma ferramenta apenas para trabalhar com tarefas (*issues*), planejamento (*kanban*), atribuição de ações, documentação, código fonte e versionamento para todas as fases do ciclo de vida de um produto de software se mostrou de grande valia. No ambiente de trabalho do laboratório trouxe como benefícios a substituição de ferramentas como draw.io, redmine e gitbook. No ambiente de residência de software, tanto presencial como online, o *onboarding* de residentes ficou simplificado com a necessidade de aprendizado de uma ferramenta e um controle melhor sobre onde os artefatos de projeto são armazenados, trabalhados e compartilhados.

V. Conclusões

A adoção de uma ferramenta declarativa para construção de modelos em UML integrada a um sistema de versionamento mostrou resultados experimentais interessantes tanto para a atuação no dia a dia de um laboratório como em projetos de extensão e em disciplinas de cursos de graduação.

Trabalhar diagramas com alunos de graduação de forma declarativa foi uma experiência que corroborou a efetividade do uso de uma ferramenta de diagramação orientada ao desenvolvedor e não ao designer (*diagramming dev tool* x *design tool)* [3].

Os resultados apresentados indicam um caminho promissor para o trabalho em projetos e disciplinas com alunos de graduação. Há ainda muitos pontos para explorar como o uso de IA neste processo de comunicação e na verificação de implementações existentes através da visualização dos modelos UML, sem entrar em detalhes da linguagem de programação utilizada, conduzindo os alunos na construção de um pensamento crítico sobre o sistema de software em construção e se afastando um pouco dos detalhes de implementação de uma linguagem de programação específica.

VI. Referencias

[1] A. Conrardy y J. Cabot, “From Image to UML: First Results of Image Based UML Diagram Generation Using LLMs,” arXiv, arXiv:2404.11376, 2024, doi: 10.48550/ARXIV.2404.11376.

[2] W. S. Cunha, H. Costa, y P. A. Parreira Júnior, “Análise de Ferramentas CASE quanto às Boas Práticas de Modelagem de Software com UML,” en *XV Simpósio Brasileiro de Qualidade de Software / XV Brazilian Symposium on Software Quality Artigos Técnicos / Research Papers*, 2016, pp. 51–63, doi: 10.5753/sbqs.2016.

[3] D2. (2025). *D2 Declarative Diagramming* [Online]. Disponible en: https://d2lang.com/tour/experience

[4] M. A. Eliseo, M. Moreira Gois, F. Silva Lopes, y I. C. Alcantara De Oliveira, “Problem-Based Learning Applied to Software Engineering: An Experience Report of The Software Residence,” en *Proceedings of the 18th Latin American Conference on Learning Technologies (LACLO 2023)*, S. Berrezueta, Ed., Singapur: Springer Nature Singapore, 2023, pp. 131–144, doi: 10.1007/978-981-99-7353-8_11.

[5] H. G. Fill, P. Fettke, y J. Köpke, “Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT,” *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, vol. 18, n.º 3, pp. 3:1–15, Abr. 2023, doi: 10.18417/EMISA.18.3.

[6] A. Koç, A. M. Erdoğan, Y. Barjakly, y S. Peker, “UML Diagrams in Software Engineering Research: A Systematic Literature Review,” en *The 7th International Management Information Systems Conference*, Mar. 2021, p. 13, doi: 10.3390/proceedings2021074013.

[7] P. Kruchten, “Architectural Blueprints: The 4+1 View Model of Software Architecture,” *IEEE Software*, pp. 42–50, 1995, doi: 10.48550/ARXIV.2006.04975.

[8] F. S. Lopes y M. A. Eliseo, “Software Residency Practices as a Complement to the Teaching-Learning Process in Software Engineering: An Experience Report,” en *2022 XVII Latin American Conference on Learning Technologies (LACLO)*, Armenia, Colombia: IEEE, Oct. 2022, pp. 1–6, doi: 10.1109/LACLO56648.2022.10013402.

[9] OMG. (2025). *Unified Modeling Language* [Online]. Disponible en: https://www.uml.org/

[10] J. Romeo, M. Raglianti, C. Nagy, y M. Lanza, “Capturing and Understanding the Drift Between Design, Implementation, and Documentation,” en *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*, Lisboa, Portugal: ACM, Abr. 2024, pp. 382–386, doi: 10.1145/3643916.3644399.

[11] PlantUML. (2025). *GitHub do PlantUML* [Online]. Disponible en: https://github.com/plantuml/plantuml

[12] E. C. Souza, F. K. Takase, R. L. Costa, y F. S. Aguchiku, “Toward Systematic Software Reuse: From Concept to Modular Software Implementation,” en *Advances in Transdisciplinary Engineering*, IOS Press, 2016, doi: 10.3233/978-1-61499-703-0-818.

[13] M. T. Valente, *Engenharia de Software Moderna*. Independente, 2022.