CBIE 2023
CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO
Uma escola para o futuro: Tecnologia e conectividade a serviço da educação

Uma Unidade Instrucional para Apoio ao Ensino de Integração Contínua em Cursos de Graduação em Tecnologia da Informação

Osmar de Oliveira Braz Junior[1,2,3], Richard Henrique de Souza[2,3], Jean C. R. Hauck[2]

[1]Universidade do Estado de Santa Catarina (UDESC)
[2]Universidade Federal de Santa Catarina (UFSC) – PPGCC/INE/CTC
[3]Universidade do Sul de Santa Catarina (UNISUL)
Florianópolis, Santa Catarina, Brasil.

osmar.braz@udesc.br, richard.henrique@ufsc.br, jean.hauck@ufsc.br

Abstract. *Continuous Integration (CI) is a software development practice that involves the frequent and automated integration of source code into a central repository and has been widely adopted by software development organizations. However, the teaching of CI needs to be more comprehensively addressed in higher education programs in Information Technology (IT) curricula, leading to a significant research gap in this context. Therefore, this work presents the development of an Instructional Unit (IU) to assist in teaching CI in higher education IT courses. The IU is systematically developed and applied through case studies and divided into four sequential parts. Preliminary results obtained from the application of the IU suggest that students have acquired CI-related knowledge through this approach, providing a positive learning experience.*

Resumo. *A Integração Contínua (CI) é uma prática de desenvolvimento de software que envolve a integração frequente e automatizada do código-fonte em um repositório central, tem sido amplamente adotada pelas organizações de desenvolvimento de software. No entanto, o ensino da CI não costuma ser abordado de maneira abrangente nos currículos dos cursos superiores na área de Tecnologia da Informação (TI), e há uma lacuna de pesquisa significativa nesse contexto. Assim, este trabalho apresenta o desenvolvimento de uma Unidade Instrucional (UI) destinada a auxiliar o ensino da CI em disciplinas de cursos superiores de TI. A UI é sistematicamente desenvolvida e aplicada por meio de estudos de caso, sendo dividida em quatro partes sequenciais. Os resultados preliminares obtidos na aplicação da UI sugerem que os alunos adquiriram conhecimentos relacionados à CI por meio dessa abordagem, proporcionando uma experiência de aprendizagem positiva.*

1. Introdução

A entrega de novas versões dos produtos de software o mais rápido possível tem sido considerada um dos principais desafios para profissionais da área de desenvolvimento de software[Humble and Farley 2013, Fontoura 2019, Shafiee et al. 2020]. Nesse sentido, a Integração Contínua (do inglês, *Continuous Integration* - CI) é uma prática de desenvolvimento de software que envolve a integração frequente e automatizada do código-fonte em um repositório central, combinada com a execução de testes automatizados [Fowler and Foemmel 2006].

A CI envolve a automação dos processos de compilação, testes e implantação do software, de forma a reduzir os riscos e garantir a qualidade do código produzido [Fowler and Foemmel 2006]. A CI é baseada em três pilares fundamentais: (i) integração frequente: os desenvolvedores integram seu trabalho ao repositório principal com frequência, várias vezes ao dia, em vez de esperar até o final de uma iteração de desenvolvimento; (ii) automação de compilação, testes e implantação, e; (iii) *feedback* rápido: a automação desses processos permite que os desenvolvedores recebam *feedback* imediato sobre a integração do seu trabalho com o código existente e possíveis problemas. A CI proporciona benefícios como a redução de erros de integração, a detecção precoce de problemas e a possibilidade de lançamentos frequentes e seguros do software. A CI também promove a colaboração e a comunicação contínua entre os membros da equipe de desenvolvimento [Fowler and Foemmel 2006] [Silva and Bezerra 2022].

Desta forma, tendo em vista que há uma tendência em se utilizar CI em conjunto com métodos ágeis [Júnior et al. 2022], amplamente utilizados na indústria de software, é importante que estudantes de cursos superiores das áreas de Tecnologia da Informação (TI) aprendam a usar CI, além de uma determinada linguagem de programação, a fim de aprenderem a desenvolver software com qualidade.

No entanto, o aprendizado de CI pode ser algo desafiador para alunos iniciantes, pois introduz complexidades devido à curva de aprendizado das ferramentas envolvidas e ao tempo disponível para o ensino desses tópicos [Eddy et al. 2017]. Apesar da importância do tópico de CI no ensino de computação, são poucas ainda as pesquisas sobre práticas de ensino eficazes para incorporar conceitos de CI em cursos de graduação nas áreas da computação [Bowyer and Hughes 2006].

Assim, este trabalho apresenta o desenvolvimento e aplicação de uma unidade instrucional para apoio ao ensino de CI em cursos de graduação em TI. A unidade instrucional é sistematicamente desenvolvida e aplicada em uma série de estudos de casos em sala de aula. As principais contribuições deste trabalho são (i) uma unidade instrucional desenvolvida para um tema tipicamente não contemplado nos currículos e, (ii) as experiências e resultados relatados do seu uso.

Este artigo está organizado como se segue. A Seção 2 discute os trabalhos relacionados. A Seção 3 descreve os métodos utilizados. A Seção 4 descreve a abordagem proposta. A Seção 5 apresenta a avaliação do estudo. Finalmente, a Seção 6 enumera contribuições e os trabalhos futuros.

2. Trabalhos relacionados

Apesar da importância da CI para o desenvolvimento de software atualmente, ainda são poucas as pesquisas relacionadas a melhores práticas ou ensino desses conceitos nos cursos de graduação.

Em relação ao conteúdo sobre CI que pode ser relevante para os profissionais, em [Júnior et al. 2022], os autores analisam as práticas de CI e indicam que a adoção e aderência da CI depende das características da organização. Organizações podem realizar práticas de diferentes estágios enquanto evoluem do desenvolvimento tradicional ao desenvolvimento contínuo. O Trabalho de [Silva and Bezerra 2022] procura evidenciar os problemas de não seguir corretamente a CI. Neste sentido, listam algumas das máspráticas que devem ser evitadas, tais como: falta de testes para ambiente de produção,

ramos divergentes, mau uso do hardware do servidor de CI, casos de teste não organizados e baixa cobertura de testes.

Em relação ao ensino de CI propriamente dita, o trabalho de [Schoeffel 2021], apresenta um jogo não digital e colaborativo. Dentre os conteúdos reforçados no jogo está a CI por meio de algumas atividades, onde é enfatizada a sua importância. Já nos estudos [Eddy et al. 2017], [Krusche and Alperowitz 2014] e [Christensen 2016] são apresentados diferentes materiais, técnicas e ferramentas em apoio a CI em ambiente educacional com resultados positivos. Apesar de relevantes, esses estudos abordam somente alguns aspectos independentes de CI, não cobrindo todo o ciclo de CI em uma unidade instrucional.

Assim, não foi possível encontrar na literatura unidades instrucionais especificamente direcionadas ao ensino do ciclo completo de CI para cursos de graduação nas áreas de TI.

3. Métodos

Este trabalho tem como objetivo o desenvolvimento e aplicação de uma Unidade Instrucional (UI) para apoio ao ensino de CI em disciplinas de cursos superiores em TI. Para atingir esse objetivo, foi adotada a abordagem metodológica apresentada na Figura 1

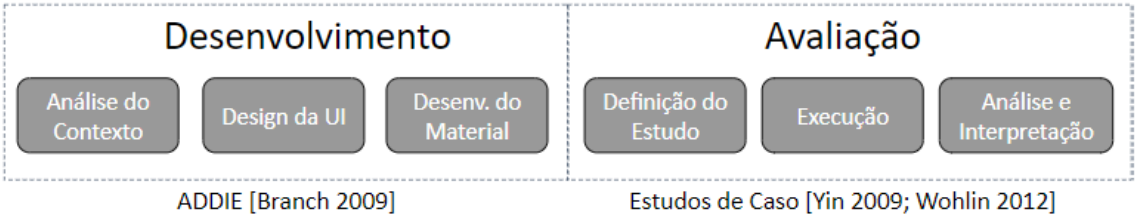

Figura 1. Abordagem metodológica adotada

Para o desenvolvimento da UI foi utilizada a abordagem ADDIE [Branch 2009]. Nesta abordagem, os alunos e o ambiente instrucional são caracterizados, as necessidades de aprendizagem são eliciadas e os objetivos de aprendizagem são definidos. A partir da análise do contexto, a estratégia instrucional é definida em termos de conteúdo, sequência e métodos instrucionais a serem adotados. O material instrucional é então desenvolvido de acordo com a estratégia instrucional.

Para avaliar a UI desenvolvida, uma série de estudos de caso seguindo as abordagens propostas por [Yin 2009] e [Wohlin et al. 2012] é realizada em sala de aula com alunos de disciplinas relacionadas à engenharia de software de diferentes cursos superiores de TI. Assim, a avaliação da UI envolve a definição, execução do estudo e análise e interpretação dos dados coletados, utilizando uma estratégia de pós-teste. A estratégia de somente pós-teste foi adotada pois a interação inicial com os estudantes revelou que nenhum dos participantes dos estudos de caso tinha conhecimento ou experiência prévia com o conteúdo de CI.

4. Unidade Instrucional Proposta

4.1. Contexto e Design da UI

No intuito de atender à necessidade de apoiar o ensino de CI em disciplinas relacionadas à engenharia de software de cursos de TI, é desenvolvida uma UI. Como o conteúdo de CI

possivelmente não será o conteúdo principal das disciplinas nas quais esta UI será inserida, a UI foi projetada para ter 12 horas/aula de duração, podendo assim ser introduzida em disciplinas onde este conteúdo de CI seja entendido como relevante. Como materiais didáticos, são desenvolvidos um conjunto de slides, um *checklist*, um roteiro detalhado para execução das atividades práticas.

O contexto da UI são disciplinas de cursos de graduação da área de TI, especialmente disciplinas relacionadas à engenharia de software ou avançadas de programação. O público-alvo da unidade instrucional são estudantes de cursos de graduação da área de TI, a partir da segunda fase, que já tenham cursado disciplinas básicas de programação. Assim, os estudantes já possuem conhecimento e prática com a sintaxe básica da linguagem utilizada. A UI será aplicada por professores de disciplinas de nível de graduação relacionadas à engenharia de software ou programação. A Tabela 1 apresenta o conteúdo programático da unidade instrucional desenvolvida.

A Figura 2 apresenta a relação dos conceitos de CI abordados: Desenvolvimento Ágil, Integração, Entrega e *Deploy* Contínuos e os conteúdos da UI proposta. As letras representam as etapas da UI de acordo com os 4 conteúdos programáticos na Tabela 1.

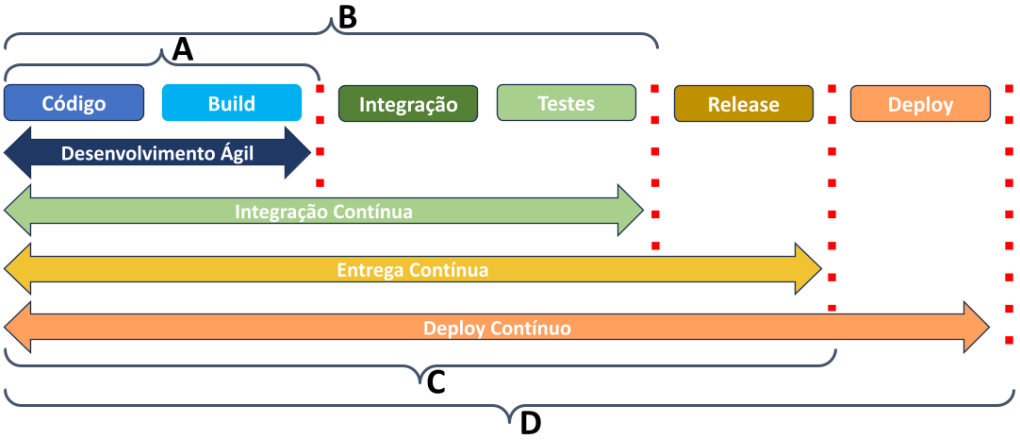

Figura 2. Etapas da UI proposta relacionados com CI

4.2. Estratégia Instrucional

A estratégia instrucional da UI se baseia na realização de um roteiro de passos de forma a executar um exemplo de CI de uma aplicação desktop da análise até a cobertura do código. A CI é realizada em 3 ambientes distintos com tarefas distintas. A análise considera as diversas métricas de qualidade de software como confiabilidade, manutebilidade, segurança, cobertura e duplicação de código.

O desenvolvimento das atividades é realizado em *Pair Programming*, de forma a reforçar o aprendizado dos métodos ágeis [Williams 2001]. Desta forma, um aluno é o piloto, responsável por escrever o código, o outro o navegador, o qual acompanha a escrita de código e verifica se está de acordo com os padrões do projeto.

Para a execução do roteiro é necessário uma IDE com suporte a Apache Maven[1],

[1] https://maven.apache.org/

Tabela 1. Conteúdo programático

Conteúdo	Método instrucional	Recursos	Carga horária
A. Criação de Projeto e testes unitários	- Apresentação expositiva/dialogada - Atividade prática seguindo a apresentação passo a passo do professor - Criar projeto na IDE - Configurar a automação com Apache Maven - Criar testes unitários com JUnit 4 - Armazenar projeto no GitHub - Discutir os resultados da etapa	- Roteiro - *Checklist* - IDE - Computadores - Projetor	3 h/a
B. Integração Continua	- Apresentação expositiva/dialogada - Atividade prática seguindo a apresentação passo a passo do professor - Configurar o GitHub Actions - Configurar o JUnit 4 para ser utilizado nas ferramentas de automação - Discutir os resultados da etapa	- Roteiro - IDE - Computadores -Projetor	3 h/a
C. Análise do Código	- Apresentação expositiva/dialogada - Atividade prática seguindo a apresentação passo a passo do professor - Configurar o SonarCloud - Realizar a integração com GitHub Actions - Discutir os resultados da etapa	- Roteiro - IDE - Computadores - Projetor	3 h/a
D. Cobertura do Código	- Apresentação expositiva/dialogada - Atividade prática seguindo a apresentação passo a passo do professor - Configurar o JaCoCo - Integração com GitHub Actions - Integração com Maven e SonarCloud - Discutir os resultados da etapa	- Roteiro - IDE - Computadores - Projetor	3 h/a

JUnit 4[2] e GitHub[3]. O aluno também necessita criar uma conta no GitHub e configurar a integração com GitHub Actions[4], SonarCloud [5] e Java Code Coverage (JaCoCo) [6]. As subseções a seguir apresenta cada etapa da UI proposta com mais detalhes.

4.3. Etapa A

Antes de iniciar, é necessário escolher uma IDE com suporte as tecnologias utilizadas[7], como por exemplo: Eclipse, Netbeans, BlueJ, JDeveloper, IntelliJ IDEA, JCreator e DrJava dentre outros. O primeiro passo do roteiro é criar um projeto na IDE utilizando o padrão de automatização do Apache Maven. A ideia aqui é ter o arquivo de configuração "pom.xml". Em seguida (passo 2) o aluno deve criar o pacote e a classe, no exemplo proposto é o pacote calculadora e a classe "Calculadora". Para executar o programa o aluno deve criar a classe Principal (passo 3), a qual utilizará a classe "Calculadora".

No passo 4, é definida uma checagem, onde o aluno é instruído a compilar e executar o código, além de refletir o que foi realizado até agora. Neste ponto aproveita-

[2] https://junit.org/junit4/
[3] https://github.com/
[4] https://github.com/features/actions/
[5] https://sonarcloud.io/
[6] https://www.eclemma.org/jacoco/
[7] https://www.geeksforgeeks.org/7-best-java-ide-for-developers-in-2022/

se para sincronizar as etapas com todos os alunos da turma. Continuando (passo 5), os alunos devem criar um repositório no GitHub (aqui é realizada uma pequena explicação básica dos comandos Git, como *commit* e o *push*). Também é necessário adicionar os colaboradores no projeto (passo 6), para a dupla ter acesso ao repositório do projeto.

A sincronização do projeto no GitHub é o passo 7. Aqui a ideia é que ambos os alunos tenham o projeto funcionado. Dependendo da IDE, pode-se optar para configurar um plug-in da respectiva IDE para acesso ao GitHub. Também pode ser visto como um ponto de controle por parte do professor, caso ache necessário sincronizar a turma antes de continuar.

Neste momento (passo 8) inicia-se a elaboração dos testes unitários, neste caso é necessário criar a classe "TestCalculadora". Em seguida escrever o método "testGetAdicao" e executar o teste (Passo 9). Se o aluno seguiu o roteiro corretamente o teste deve informar sucesso, sem erros. Então como passo 10, o aluno deve analisar o teste realizado. Aqui é utilizado o *checklist* para o aluno poder verificar o que ocorreu de errado.

Depois de ter o primeiro teste unitário funcionando, chegou a hora de criar a classe "TestSuite" (Passo 11). Além disso é solicitado aos alunos conferirem se a estrutura de diretórios está em conformidade ao solicitado até o momento, muito importante para que a automação funcione corretamente. Agora (Passo 12), os alunos devem executar o teste novamente usando a classe "TestSuite".

Para finalizar a etapa A (Passo 13), os alunos devem submeter as alterações para o GitHub. Momento importante para o professor sincronizar a turma antes de ir para a etapa B. Caso a Etapa B não seja realizada no mesmo dia, sugere-se fazer uma revisão dos conceitos abordados e verificar se todos terminaram de forma adequada.

4.4. Etapa B

Nesta etapa, são realizadas as configurações do GitHub Actions para viabilizar a utilização do SonarCloud. No Passo 1, é solicitado aos alunos que criem o *workflow* para a integração contínua no GitHub Action com Java e Maven. Em seguida (Passo 2), o arquivo "maven.yml" é modificado para configurar o nome do *workflow*, o evento que desencadeia o *workflow*, a versão do sistema operacional a ser utilizada, a versão do Java e, por fim, o comando para executar a tarefa conforme especificado no arquivo "pom.xml".

Após efetuarem as alterações requeridas, os alunos devem proceder com o *commit* no GitHub (Passo 3). Dessa forma, ao submeterem a modificação do *workflow*, este será executado (Passo 4). É responsabilidade dos alunos monitorar a execução, uma vez que os jobs de construção (*build*) do *workflow* são realizados pelo GitHub Actions, utilizando a automação fornecida pelo Apache Maven.

O Passo 5 é o momento em que as equipes (duplas) devem sincronizar seus projetos locais. Esse é um momento excelente para o professor, se desejar, acompanhar e sincronizar o progresso do roteiro com a turma como um todo. Logo em seguida, são adicionados mais testes, especificamente o teste de subtração ("testeGetSubtracao"). Após a elaboração do teste, os alunos são orientados a executá-los (Passo 6), os quais devem ser bem-sucedidos. Os alunos devem então submeter as alterações realizadas ao repositório (Passo 7) e monitorar a execução do *workflow* no GitHub Actions.

Agora, com tudo funcionando, o Passo 9 é a criação de 3 ambientes (do inglês,

enviroments): (i) o ambiente desenvolvimento (dev) para desenvolvimento e compilação do projeto, (ii) o ambiente homologação (hmg) para os testes unitários e métricas de software, e o (iii) ambiente produção (prd) para o empacotamento do projeto.

Em seguida (passo 10) os alunos devem configurar o arquivo pom.xml para que empacote o projeto com as dependências (i.e., *jar-with-dependencies*). Também é solicitado (Passo 11) para ajustar o arquivo "maven.yml" para executar as tarefas em cada um dos ambientes criados. Então (Passo 12) os alunos devem submeter as alterações dos arquivos "pom.xml" e "maven.yml" ao repositório GitHub.

Por fim (Passo 13), os alunos devem verificar se o *workflow* da integração continua foi executado com sucesso. Momento de sincronizar a turma, fomentar a discussão do que foi realizado até o momento.

4.5. Etapa C

Nesta Etapa é realizada a integração do SonarCloud com GitHub Actions. Sendo assim o Passo 1 é a criação da conta no SonarCloud. Em seguida (Passo 2), os alunos devem criar uma organização no SonarCloud vinculada ao GitHub. Também (Passo 3) deve-se configurar o SonarCloud para ter acesso (senha) ao repositório do projeto. O Passo 4 é criar um projeto SonarCloud e selecionar o projeto no GitHub a ser vinculado ao projeto no SonarCloud (Passo 5).

No passo 6, os alunos devem configurar o SonarCloud para recuperar as configurações do projeto (i.e., GitHub actions). Como a análise é realizada via GitHub Actions o mesmo deve ser configurado no SonarCloud (Passo 7). Da mesma forma a configuração no GitHub deve ser realizada para que o SonarCloud possa acessar o projeto (Passo 8).

Em seguida (Passo 9) os alunos devem visualizar as alterações nos arquivos "pom.xml" e "maven.yml". Então (Passo 10) inserir os dados do SonarCloud no arquivo "pom.xml". Também adicionar a configuração necessária na tag "*properties*" do arquivo "pom.xml" (Passo 11).

Agora (Passo 12), os alunos devem modificar o arquivo "maven.yml" inserindo as alterações necessárias. Voltando ao SonarCloud (Passo 13), os alunos devem conferir as chaves do projeto (do inglês, *Project keys*). Assim, os alunos poderão ver a análise do projeto no SonarCloud.

Falta então submeter (Passo 14) as alterações dos arquivos "pom.xml" e "maven.yml" ao repositório GitHub. Em seguida (Passo 15), deve-se verificar se o *workflow* da integração continua foi executada com sucesso. Finalmente os alunos podem acessar a análise do código do repositório no SonarCloud (Passo 16).

No passo 17, os alunos devem criar o arquivo "README.md" na raiz do projeto para criar *badges* no repositório. Em seguida ser submetido ao repositório (Passo 18). Agora (Passo 19), os alunos podem acessar o repositório e ver como ficou o projeto com o descrito do arquivo "README.md".

4.6. Etapa D

Nesta etapa é adicionado o JaCoCo para a análise do projeto. Então, o passo 1 é adicionar o plugin do JaCoCo ao arquivo "pom.xml". Em seguida (passo 2), os alunos devem alterar arquivo "maven.yml" para enviar o arquivo do relatório do JaCoCo para o Sonar.

Com as devidas alterações é hora de submeter ao repositório GitHub as alterações dos arquivos "pom.xml" e "maven.yml" (Passo 3). Em seguida, pode-se verificar as informações de cobertura (do inglês, *coverage*) do código (Passo 4) e pode-se também ver a análise do último *commit* (*new code*). O professor revisa essas medidas juntamente com os alunos, promovendo uma discussão sobre o significado de cada uma.

Ao acessar o GitHub (Passo 5), pode-se ver o percentual de cobertura. Se tudo estiver correto, agora pode-se finalizar o projeto como um todo, para conseguir o 100% de cobertura. Então após a execução dos roteiros é proposto aos alunos: Aumentar a cobertura do código com testes para os outros métodos da classe Calculadora, avaliar o resultado da execução do *workflow* e avaliar o resultado da análise do Sonar.

Através da implementação dos roteiros, almeja-se que os estudantes adquiram a compreensão de que a CI constitui um processo fundamental para a sustentabilidade de qualquer software ao longo de um período de tempo prolongado. Adicionalmente, destaca-se que o conhecimento e a proficiência nas ferramentas pertinentes são elementos críticos para assegurar a eficiência no processo de distribuição de software.

Disponibilizamos os materiais instrucionais (i.e., slides e pdf) e os códigos fontes em: "`https://github.com/osmarbraz/calculadora_sbie2023`".

5. Avaliação

A UI desenvolvida é avaliada por meio de uma série de estudos de caso aplicados em disciplinas de cursos superiores da área de TI. Seguindo a abordagem metodológica, os estudos são definidos, executados e os dados são coletados, analisados e interpretados.

5.1. Definição dos estudos

Os estudos aplicados têm o objetivo de avaliar a aprendizagem de CI sob o ponto de vista dos estudantes que participaram das aulas utilizando a UI desenvolvida.

Com base no objetivo desta pesquisa, são definidas duas questões de análise para o estudo:

1. Como foi o aprendizado dos conceitos de CI?
2. Como foi a experiência de aprendizagem?

A amostragem dos estudos é realizada de forma não-probabilística por conveniência, com base nos estudantes matriculados nas disciplinas. A UI é aplicada pelos próprios autores, que são professores das disciplinas.

A intervenção realizada consiste na aplicação da UI desenvolvida em sala de aula e em laboratório, agregada à parte final das disciplinas.

A coleta de dados se dá por observação participativa e pós-teste aplicado ao final da disciplina. Para a definição do pós-teste foi aplicada uma prova aos alunos, questionando os conhecimentos adquiridos nos conteúdos relacionados a CI.

5.2. Execução dos estudos

Os estudos foram aplicados durante o primeiro e segundo semestres de 2022 nas turmas de Modelos Métodos e Técnicas de Engenharia de Software dos cursos de Ciência da

Computação, Sistemas de Informação, Análise e Desenvolvimento de Sistemas e Gestão de TI da Universidade do Sul de Santa Catarina (UNISUL).

Participam dos estudos 119 estudantes matriculados nas seis turmas das disciplinas. Para a execução da UI, foram destinadas 4 aulas, cada uma com duração de 3 horas, totalizando 12 horas de atividades. Desse tempo, duas aulas foram dedicadas à parte teórica, com explanação dos conceitos envolvidos, e as outras duas aulas foram voltadas para a aplicação prática da UI.

5.3. Resultados e Discussão

Nesta seção os dados coletados são analisados e discutidos.

Como foi o aprendizado dos conceitos de CI?

Após a aplicação do pós-teste, foi possível observar que 83,2% dos estudantes participantes obtiveram média acima de 7,0 nos conteúdos de CI da prova aplicada. Esse dado levanta indícios de que a aplicação da UI foi realizada com sucesso em relação à aprendizagem dos conteúdos.

Como foi a experiência de aprendizagem?

Durante a aplicação da UI foram coletadas diversas impressões com base em relatos textuais dos alunos e na observação participante dos professores.

Todos os alunos participantes conseguiram concluir com sucesso todas as etapas propostas na UI. A maioria dos alunos concluiu o roteiro de forma entusiasmada e relatou a relevância do tema para suas futuras carreiras. Essa percepção foi evidenciada pelo fato de os alunos terem realizado as atividades práticas durante a implementação da UI, bem como pelos relatos dos alunos durante e após sua execução.

Foi possível observar que os estudantes mantêm o foco na execução das atividades práticas da UI, uma vez que os conceitos relevantes foram abordados previamente em aulas expositivas. Isso cria um ambiente propício para o esclarecimento de dúvidas à medida que os estudantes avançam na execução do roteiro, garantindo uma boa experiência de aprendizado.

Durante a implementação da UI, pôde-se também observar uma distinção entre os estudantes que já possuem experiência profissional na área de TI, apesar de não terem prática com CI, e aqueles que ainda não têm vivência profissional. Os estudantes que não possuem experiência profissional inicialmente demonstraram dificuldades em compreender a relevância da aplicação da CI, enquanto os que já trabalham na área de TI não apenas possuem uma compreensão mais sólida do motivo pelo qual essa prática é utilizada, mas também trazem consigo suas experiências do ambiente de trabalho.

Outro aspecto relatados pelos alunos foi a habilidade de destacar as más práticas presentes em seus respectivos ambientes de trabalho, corroborando de forma semelhante ao descrito por [Silva and Bezerra 2022]. Essa constatação gerou uma discussão enriquecedora em sala de aula. Adicionalmente, os estudantes também foram capazes de identificar as boas práticas que são aplicadas em seus contextos laborais.

Alguns estudantes enfrentaram desafios durante a implementação da UI. Uma das dificuldades encontradas pelos alunos, especialmente aqueles sem experiência profissional prévia, foi a utilização de um repositório (i.e., GitHub). Como resultado imediato, a

UI foi complementada com um material adicional sobre o uso adequado do repositório, incluindo indicações de vídeos e tutoriais.

Outra dificuldade enfrentada pelos estudantes foi a integração e configuração de diferentes tecnologias, mesmo com o roteiro fornecendo detalhes sobre o passo-a-passo necessário. No entanto, essa dificuldade, uma vez vencida, foi relatada pelos estudantes como um aspecto positivo em termos de preparação dos alunos para o mercado.

No geral, a percepção dos professores foi positiva. Isso se deve, em grande parte, ao fato de que a maioria dos estudantes não tinha experiência prévia com a utilização de um sistema de controle de versão para o seus projetos de software. Além disso, a oportunidade de usar testes automatizados e entregar software de forma clara aos clientes (usuários finais) foi impactante e inovadora. Isso aprimorou a preparação dos alunos para desafios no desenvolvimento de software.

5.4. Ameaças à validade

Algumas possíveis ameaças à validade foram observadas neste estudo. Foram adotadas estratégias para minimizar o impacto dessas ameaças. Ameaças à validade de construção estão relacionadas à estratégia de coleta de dados, que foi baseada somente em pós-teste e em observação participante. Essa ameaça foi mitigada pela revisão cruzada criteriosa entre os professores participantes acerca das observações coletadas. Ameaças à validade externa podem ocorrer pela estratégia de seleção e tamanho da amostra. Esta ameaça foi mitigada pelo envolvimento de dois professores e mais de 100 alunos de seis turmas de diferentes disciplinas. No entanto, estudos com amostragem mais ampla ou grupos de controle podem ser necessários para confirmar os achados.

6. Conclusões e trabalhos futuros

Este trabalho apresenta o desenvolvimento e aplicação de uma Unidade Instrucional (UI) para o ensino de Integração Contínua (CI) em disciplinas de áreas relacionadas à engenharia de software ou programação. A UI é sistematicamente desenvolvida e aplicada em uma série e estudos de caso. A CI é uma importante técnica para garantir a entrega aos clientes de um produto de software de qualidade. Contudo a falta de conhecimento e experiência pode ser fator de dificuldade da sua adoção pelas empresas. Dessa forma, é importante que o ensino desses conteúdos seja incorporado às disciplinas de cursos de TI.

Desta forma, a aplicação da UI para ensino de CI se mostrou eficaz para a aprendizagem dos estudantes. De maneira a melhorar a compreensão dos benefícios do uso da CI no desenvolvimento de software. A experiência de aprendizagem relatada pelos estudantes na execução das atividades indicam que eles compreenderam a importância dos conceitos e práticas apresentadas na disciplina. Além disso, o repositório criado durante as atividades pode ser agregado ao portfólio de cada um dos estudantes.

Trabalhos futuros incluem desenvolver a UI considerando outras linguagens de programação (e.g., Python) e explorar métricas de qualidade de software. Além disso, a aplicação da UI em grupos maiores de alunos, ou utilizando experimentos com grupos de controle, pode ampliar a avaliação da UI proposta.

Referências

Bowyer, J. and Hughes, J. (2006). Assessing undergraduate experience of continuous integration and test-driven development. In *Proceedings of the 28th international conference on Software engineering*, pages 691–694.

Branch, R. M. (2009). *Instructional design: The ADDIE approach*, volume 722. Springer.

Christensen, H. B. (2016). Teaching devops and cloud computing using a cognitive apprenticeship and story-telling approach. In *Proceedings of the 2016 ACM conference on innovation and technology in computer science education*, pages 174–179.

Eddy, B. P., Wilde, N., Cooper, N. A., Mishra, B., Gamboa, V. S., Shah, K. M., Deleon, A. M., and Shields, N. A. (2017). A pilot study on introducing continuous integration and delivery into undergraduate software engineering courses. In *2017 ieee 30th conference on software engineering education and training (csee&t)*, pages 47–56. IEEE.

Fontoura, F. C. (2019). Uso de metodologias de desenvolvimento de software e de engenharia de requisitos em empresas de tecnologia: um estudo a partir de um survey. B.S. thesis, Universidade Federal do Rio Grande do Norte.

Fowler, M. and Foemmel, M. (2006). Continuous integration.

Humble, J. and Farley, D. (2013). Entrega contínua: como entregar software de forma rápida e confiável.

Júnior, P. S. S., Barcellos, M. P., Ruy, F. B., and Omêna, M. S. (2022). Flying over brazilian organizations with zeppelin: A preliminary panoramic picture of continuous software engineering. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering*, pages 279–288.

Krusche, S. and Alperowitz, L. (2014). Introduction of continuous delivery in multi-customer project courses. In *Companion Proceedings of the 36th International Conference on Software Engineering*, pages 335–343.

Schoeffel, P. (2021). Xp enigma-um jogo educacional não digital para apoio ao ensino de métodos ágeis: Uma análise temporal da motivação e aprendizagem. In *Anais do Simpósio Brasileiro de Educação em Computação*, pages 152–161. SBC.

Shafiee, S., Wautelet, Y., Hvam, L., Sandrin, E., and Forza, C. (2020). Scrum versus rational unified process in facing the main challenges of product configuration systems development. *Journal of Systems and Software*, 170:110732.

Silva, R. B. T. and Bezerra, C. (2022). Empirical investigation of the influence of continuous integration bad practices on software quality. In *Anais do X Workshop de Visualização, Evolução e Manutenção de Software*, pages 51–55. SBC.

Williams, L. (2001). Integrating pair programming into a software development process. In *Proceedings 14th Conference on Software Engineering Education and Training.'In search of a software engineering profession'(Cat. No. PR01059)*, pages 27–36. IEEE.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Yin, R. K. (2009). *Case study research: Design and methods*, volume 5. sage.