Projeto e Avaliação de um Template de *Worked Examples* para o Ensino de Programação

Andre L. M. Miranda[1], **Renato Garcia** [1,2], **Gabriel M. Lunardi**[3], **Ricardo Vilela**[4], **Pedro H. D. Vale**[5], **Williamson Silva**[1,2]

[1] Universidade Federal do Pampa - UNIPAMPA (Campus Alegrete), Alegrete, RS, Brasil

[2]PPGES (UNIPAMPA - Campus Alegrete), Alegrete, RS, Brasil

[3]Universidade Federal de Santa Maria – UFSM, Cachoeira do Sul, RS, Brasil

[4]Universidade Federal do Cariri – UFCA, Juazeiro do Norte, CE, Brasil

[5]Universidade Federal de Juiz de Fora – UFJF, Juiz de Fora, MG, Brasil

[1]andremendes0113@gmail.com,[1,2]renatogarcia.aluno@unipampa.edu.br

[3]gmlunardi@gmail.com,[4]ricardovilela91@gmail.com

[5]pedrohenrique.valle@ufjf.br,[1,2]williamsonsilva@unipampa.edu.br

Abstract. *Worked Examples (WE) have become increasingly popular in teaching various subjects. However, more research is needed to explore the use of WE in programming education. The lack of an educational tool to assist teachers in effectively incorporating WE may contribute to this limitation. Before implementing WE in a tool, we created a template. This work presents a template designed to help instructors standardize the use of WE in programming education. The work also includes findings from an exploratory study conducted with instructors who used the template. The study provided valuable insights into the template's feasibility and effectiveness.*

Resumo. *O uso de Worked Examples (WE) tem ganhado destaque no ensino de diferentes áreas de conhecimento. Contudo, no ensino de programação, existem poucos trabalhos explorando o assunto. A falta de uma ferramenta educacional projetada para orientar os docentes na aplicação eficaz dos WE pode ser um dos fatores que contribuem para essa limitação. Contudo, para implementar os WE em uma ferramenta é necessário primeiro desenvolver um template. Este trabalho apresenta um template projetado com o objetivo de auxiliar os docentes a padronizar o uso de WE no ensino de programação. O trabalho também descreve os resultados de um estudo exploratório, realizado com docentes que usaram o template, que forneceram insights valiosos sobre a viabilidade e a eficácia do uso do template.*

1. Introdução

A aprendizagem não se resume a um processo de absorção passiva, repetição de práticas e fortalecimento da memória [Bada and Olusegun 2015]. Em vez disso, trata-se de processo ativo de construção de significado por meio da interação entre o indivíduo e o

ambiente, levando em consideração o conhecimento prévio e a experiência dos estudantes [Bada and Olusegun 2015]. Nesse sentido, os docentes podem empregar estratégias pedagógicas com o objetivo de alcançar resultados de aprendizagem mais eficazes e eficientes. Além disso, é importante observar que, em determinados conteúdos, os estudantes devem receber orientação especial durante a fase inicial de aprendizado para desenvolver uma mentalidade de pensamento abstrato e compreender o conceito de projetar os componentes (objetos) de um sistema.

Uma estratégia que tem ganhado destaque é o uso de exemplos, por meio de *Worked Example* (WE), em sala de aula [Renkl 2014, Renkl 2017]. Ensinar a partir de WE, também conhecido como "Aprendizagem Baseada em Exemplos", é recomendado e utilizado quando se deseja promover a aquisição inicial de habilidades cognitivas, especialmente, quando os estudantes estão aprendendo a resolver certos tipos de problemas [Renkl 2017]. Essa estratégia encoraja os estudantes a explicar a lógica por trás das soluções para si mesmos [Rittle-Johnson et al. 2017, Renkl 2017]. Para isso, os WE devem incluir um problema, as etapas de processamento da solução e, por fim, uma solução final que auxilia os estudantes a desenvolverem esquemas de aprendizagem generalizados. Devido à sua eficácia, os WE são frequentemente utilizados no ensino problemas matemáticos e físicos [Adams et al. 2014, McLaren et al. 2016].

Apesar da popularidade dos WE em outras áreas de conhecimento, há uma escassez de estudos que abordam sua prática na área de Computação, especialmente no ensino de programação. *Worked Examples* podem ser facilmente adaptados para esse contexto, já que a programação pode ser dividida em linhas de código sucessivas, representando as etapas de processamento da solução, que culminam em um código mais complexo para resolver um problema. Aprender a programar envolve adquirir uma estratégia para resolver problemas, e os exemplos trabalhados têm se mostrado úteis nesse processo [van Merrienboer 1990, Hosseini et al. 2020]. No entanto, muitas vezes, os docentes limitam-se a apresentar exemplos com conceitos básicos e, gradualmente, abordam tópicos mais complexos. Sem o uso de esquemas ou estruturas adequadas que permitam aos estudantes explorar todo o seu potencial de aprendizagem, eles podem enfrentar dificuldades na compreensão de novos conteúdos apresentados em sala de aula. Segundo Atkinson *et al.* (2000), o design e a estrutura dos WEs são importantes ao discutir a eficácia desses recursos em sala de aula. É crucial considerar o *design* e a estrutura dos materiais de aprendizagem, a fim de otimizar a compreensão e o aprendizado dos estudantes.

Diante desse cenário, o objetivo deste trabalho é apresentar a proposta de um *template*, um guia que auxiliará os docentes na criação e no uso de WE no contexto de ensino de programação. Acreditamos que ao utilizar esse *template*, os docentes poderão adotar exemplos de forma mais eficaz em sala de aula, resultando em melhores resultados de aprendizado aos estudantes. Além da proposta do *template*, apresentamos um estudo exploratório com docentes para avaliar a viabilidade de uso e aceitação do *template*. A avaliação da aceitação foi conduzida por meio de um questionário com base nos indicadores do Modelo de Aceitação de Tecnologia (do inglês, *Technology Acceptance Model* – TAM) [Venkatesh and Davis 2000]. Em seguida, foram conduzidas entrevistas de acompanhamento com os docentes, permitindo que expressassem suas opiniões, facilidades e dificuldades em relação ao *template* e seu uso.

2. Fundamentação e trabalhos relacionados

Worked Examples (WE) estão fundamentados na Teoria da Carga Cognitiva, que defende que o ensino é aprimorado quando não há uma sobrecarga de informações, permitindo que o foco seja mantido nos pontos-chave do problema [Sweller et al. 1998]. O uso de WE auxilia na redução da carga cognitiva, fornecendo uma estrutura a ser seguida para alcançar o resultado esperado, o que beneficia a compreensão e o entendimento do conteúdo em questão. Existem diversas abordagens para realizar demonstrações de WE no processo de aprendizagem e algumas desviaram-se da apresentação de uma solução didática e usaram exemplos errôneos com uma instrução para os estudantes encontrarem e corrigirem os erros [Große and Renkl 2007], ou com *feedback* adicional [Kopp et al. 2008] para estimular os estudantes a processar os exemplos mais profundamente.

Os *worked examples* são ferramentas úteis para docentes e até mesmo para aqueles que estão aprendendo de forma autodidata. Eles podem ser moldados para se adequarem a diversas realidades educacionais. O estudo dos *worked examples* é um tema relevante para compreender e aprimorar as práticas de ensino e aprendizagem. O uso de WE pode ser evidenciado em diversas áreas do conhecimento, como Matemática [McGinn et al. 2015], Física [Cartwright and McMullin 1984], Química [Monk 2008], Computação [Abdul-Rahman and du Boulay 2014], entre outras áreas. A seguir vamos apresentar alguns trabalhos mais relevantes que utilizaram exemplos como modelo de aprendizagem para o ensino de programação.

Garces *et al.* (2023) realizaram um experimento para explorar o uso de exemplos práticos no ensino de programação. O experimento avaliou como a forma de aprendizagem pode afetar os estudantes no processo de aprendizagem, utilizando estratégias de *debugging* e *self-explaining*. Os participantes eram de duas turmas de um curso de computação e foram divididos em dois grupos: um com estudantes que já possuíam experiência em programação e outro sem experiência. Em cada turma foi aplicada uma abordagem para o uso dos exemplos. Embora as estratégias tenham sido úteis para o ensino de programação, os resultados não mostraram uma grande disparidade entre elas. No entanto, independentemente do estudante já possuir ou não experiência, a estratégia de *self-explaining* foi mais eficaz do que a de *debugging*.

Por meio de exemplos interativos de programação em Python, utilizando a ferramenta PCEX, que fornece uma forma envolvente para apresentar exemplos de construção de programas, Hosseini *et al.* (2020) compararam *worked examples* com métodos não interativos convencionais. Os autores conduziram um experimento controlado, dividindo os estudantes em dois grupos, um para cada método de exemplo. A avaliação consistiu em comparar resultados com base no engajamento, desempenho na resolução de problemas e aprendizado. Os resultados mostraram que os estudantes que tiveram contato com métodos iterativos tiveram um desempenho relativamente maior na resolução de problemas, porém levaram mais tempo para resolver os subproblemas de construção de programas. Quanto ao aprendizado, não houve diferença significativa entre os dois grupos.

Diante dos trabalhos mencionados acima, é possível identificar um interesse e produção em curso, utilizando WE para ensino e aprendizagem de programação. Esses estudos corroboram a observação de um campo ainda recente e a ser explorado. O potencial relatado por esses estudos justifica a necessidade de novas pesquisas no que se refere ao ensino e aprendizagem por meio de aplicação de *worked examples*. Outro fator

identificado neste estudo é que cada grupo de autores definiu suas próprias regras para construção um *worked examples*, o que evidencia um nicho a ser explorado em nossa pesquisa, ou seja, a proposição de um *template* que possa orientar os docentes na produção de *worked examples* para ensino de programação.

3. Metodologia para o desenvolvimento e avaliação do *template*

Para alcançar o objetivo esperado, este trabalho foi guiado por meio da metodologia *Design Science Research* (DSR) [Hevner 2007]. A DSR é uma metodologia que apoia o processo de criação, desenvolvimento e avaliação de artefatos que visam solucionar problemas práticos de interesse da comunidade. A DSR é composta por três ciclos interrelacionados: o Ciclo de Relevância, o Ciclo de Design e o Ciclo de Rigor.

O **Ciclo de Relevância** consiste na definição do problema a ser pesquisado. A motivação desta pesquisa está relacionada à percepção de que a aplicação de WE está limitada, resultando em uma aprendizagem ineficaz. Ao fornecer um *template* estruturado e padronizado, os docentes terão um guia claro sobre como utilizar WE em sala de aula. Isso ajudará os docentes a criar exemplos relevantes, abordando problemas específicos de programação, destacando as etapas de solução e fornecendo aos estudantes uma base sólida para a compreender conceitos e adquirir habilidades cognitivas.

O **Ciclo de *Design*** consiste no desenvolvimento e avaliação do artefato para o problema. Neste trabalho, o artefato é o *template* de *worked examples*. Para o desenvolvimento do *template*, realizamos uma pesquisa em busca de evidências na literatura de artigos publicados que já usaram WE, e tentamos identificar características comuns de como os WE podem ser aplicados. Como ponto de partida, decidimos utilizar o modelo proposto por Tonhão *et al.* (2021). A proposta dos autores apresenta os principais elementos de um exemplo trabalhado e informações consideradas importantes para o seu entendimento, além de informações de cunho pedagógico que são importantes para auxiliar os docentes na adoção desses exemplos. Além disso, encontramos outras evidências de trabalhos na literatura que incorporamos também na primeira versão do *template*. Vale ressaltar que não identificamos nenhum outro trabalho que estruture e forneça guidelines de como utilizar *worked examples* para o ensino de programação.

A Tabela 1 apresenta a versão inicial, bem como o mapeamento das informações coletadas de cada trabalho que identificamos. O *template* está dividido em três partes: “Dados Gerais", “Dados sobre o *Worked Example*” e “Aplicação do *Worked Example* (Corretos e Errôneos)". Na **Parte 01 (Dados Gerais)** do *template*, é local onde o docente fornecerá informações dos dados gerais sobre o curso e a origem do exemplo trabalhado. Nos Dados Gerais sobre o Curso, devem ser adicionadas informações como o nome da disciplina, o conteúdo que será ensinado, o tópico da disciplina, o subtópico da disciplina (conteúdos específicos, se houver) e, por fim, o conhecimento prévio necessário para os estudantes compreenderem o exemplo trabalhado. Nos Dados do Local do exemplo trabalhado, deve-se informar o local de onde exemplo foi retirado, como os slides de aula, de repositórios ou projetos abertos (GitHub). A **Parte 2 (Contexto do Exemplo Trabalhado)**, devem ser fornecidas informações que expliquem o contexto do exemplo para o discente, incluindo a descrição do problema, uma demonstração de resultado com dados de entrada e saída esperados, e material complementar para auxiliar no aprendizado do estudante. É importante que o material complementar contenha recursos adicionais,

Tabela 1. *Proposta de Template* para uso de WE para o ensino de programação.

Parte 01 - Dados Gerais	**Dados Gerais Sobre o Curso**	
	Título da disciplina	[Inserir título da disciplina] [Tonhão et al. 2021]
	Tópico da disciplina	[Inserir tópico da disciplina] [Tonhão et al. 2021]
	Subtópicos	[Inserir subtópicos da disciplina que serão abordados na atividade] [Tonhão et al. 2021]
	Conhecimento prévio	[Inserir conhecimentos prévios necessários para a compreensão da atividade] [Tonhão et al. 2021]
	Dados do Local do *Worked Examples*	
	Local onde foi retirado	[Inserir de onde o material para o *worked example* foi retirado] [Tonhão et al. 2021]
Parte 02 - Contexto	**Dados do *Worked Example***	
	Descrição do problema	[Inserir descrição da atividade a ser realizada, incluindo o problema a ser resolvido]
	Resultado	[inserir resultado para o problema abordado]
	Material complementar	[Inserir material de apoio para resolução do problema]
Parte 3.1 - *Worked Example* Correto	***Worked Example* Correto**	
	Reflexivo	[Inserir texto explicativo sobre o problema com worked example, o texto deve ajudar o aluno a refletir sobre o problema] [Abdul-Rahman and du Boulay 2014] [McGinn et al. 2015] [Rodiawati and Retnowati 2019] [Huang et al. 2015]
	Níveis de dificuldade	[Inserir níveis de dificuldade da atividade reflexiva, como fácil, médio ou difícil] [Huang et al. 2015]
	Etapas Corretas	[Inserir passo-a-passo para conclusão do exercício] [Rodiawati and Retnowati 2019] [Retnowati and Marissa 2018] [McGinn et al. 2015]
	Teste	[Inserir dados de testes para verificar se o resultado obtido é o mesmo que o esperado] [Abdul-Rahman and du Boulay 2014]
Parte 3.2 - *Worked Example* Errôneo	***Worked Example* Errôneo**	
	Reflexivo	[Inserir texto explicativo sobre o problema com worked example, o texto deve ajudar o aluno a refletir sobre o problema] [Abdul-Rahman and du Boulay 2014] [McGinn et al. 2015] [Rodiawati and Retnowati 2019] [Huang et al. 2015]
	Níveis de dificuldade	[Inserir níveis de dificuldade da atividade reflexiva, como fácil, médio ou difícil] [Huang et al. 2015]
	Etapas Errôneas	[Inserir passo-a-passo para conclusão do exercício] [Chen et al. 2016] [Huang et al. 2015]
	Você consegue identificar o erro? Indique a linha onde este ocorre	[Inserir uma explicação sobre porque o erro ocorre e em que parte do código/passo-a-passo está] [Garces et al. 2023]
	Teste	[Inserir dados de testes para verificar se o resultado obtido é o mesmo que o esperado] [Abdul-Rahman and du Boulay 2014]

como vídeos, sites ou documentação.

Em seguida, temos a **Parte 3 (*Worked Example* Correto)**, que demonstra a seção em que a solução do problema é apresentada por meio do WE ao estudante. Existem quatro itens que devem ser preenchidos. O primeiro é o "Reflexivo", no qual o docente deve fornecer um detalhamento sobre a lógica que o estudante deve seguir para compreender e resolver o problema. Isso permite que o estudante estimule a utilizar sua capacidade cognitiva e conecte o novo conhecimento com experiências anteriores de aprendizagem. Isso destaca a importância de ativar e integrar conhecimentos prévios para uma compreensão mais profunda e efetiva do WE [Abdul-Rahman and du Boulay 2014]. A partir dessas informações, o estudante poderá criar etapas para solucionar o problema, formando assim mapas mentais. O segundo item está relacionado ao nível de dificuldade do exemplo, no qual o docente deve classificar qual a dificuldade do exemplo. O terceiro item é responsável por identificar as Etapas Corretas da Solução. Neste item devem ser indicados os passos necessários para chegar a uma solução correta do problema, complementando o conteúdo conteúdo apresentado no item "Reflexivo". Por fim, o último item é responsável pelos "Testes", no qual o docente deve apresentar os possíveis dados de entradas para o problema, bem como as saídas esperadas. Dessa forma, o estudante pode receber um *feedback* sobre a sua solução.

A **Parte 4 (*Worked Example* Errôneo)** possui objetivos semelhantes aos apresentados na Parte 3. Contudo, há uma diferença, pois a Etapa 4 deve conter um ou mais

passos incorretos. A quantidade de erros no código deve ser determinada pelo docente, podendo ser erros sintáticos ou semânticos. O estudante terá acesso aos itens "Reflexivos" e as "Etapas Errôneas", nos quais poderá realizar uma análise comparativa e identificar o erro. Em seguida (Identificação do Erro), o estudante deverá identificar, compreender e relatar os erros encontrados. Assim, o estudante estará aprendendo e expandindo seus conhecimentos, além de obter informações sobre como evitar os erros em atividades futuras [Chen et al. 2016]. Após o estudante identificar e relatar o erro, o docente deverá explicar detalhadamente a causa e, em seguida, apresentar a proposta de solução correta do WE. O último campo é o mesmo que se repete em relação à etapa 4, no qual são realizados os testes com as entradas e as suas possíveis saídas.

Por fim, o **Ciclo de Rigor** é a etapa na qual se obtém a contribuição esperada, com base nas teorias que direcionaram a pesquisa para alcançar o artefato proposto, como a Teoria da Carga Cognitiva [Sweller et al. 1998], DSR [Hevner 2007], *Worked Examples* [Große and Renkl 2007, McLaren et al. 2016]. A contribuição oferecida neste trabalho é a criação de um *template* estruturado e padronizado que visa auxiliar os docentes na produção de WE no ensino de disciplinas de programação.

4. Estudo Exploratório

Na metodologia DSR, é esperado que em cada ciclo de *Design* sejam realizadas avaliações no artefato proposto [Hevner 2007]. Com base nessa premissa, esta seção apresenta o planejamento e a execução de um estudo exploratório conduzido para examinar a viabilidade de uso e a aceitação do *template* proposto, a partir da perspectiva de docentes.

4.1. Planejamento

Para facilitar a execução do estudo foram utilizadas as ferramentas disponibilizadas pelo *Google Workspace*. Os instrumentos elaborados compreenderam: (i) um termo de consentimento, que garantiu a confidencialidade dos dados fornecidos e o anonimato dos participantes; (ii) questionários de caracterização de perfil, para obter informações sobre os conhecimentos relacionados a exemplos e as características dos docentes; (iii) documentos contendo o roteiro do estudo e as instruções necessárias para a realização da avaliação; (iv) uma apresentação sobre *worked examples* e o *template* proposto.

4.2. Participantes

Esse estudo é de natureza exploratória e qualitativa, portanto, a representatividade da amostra é mais importante do que a quantidade. Dado esse contexto, quatro docentes do curso de Engenharia de Software, da Universidade Federal do Pampa (Unipampa), foram convidados para participar do estudo (D1, D2, D3 e D4). O convite foi realizado por conveniência. Todos os docentes tinham mais 10 de anos de experiência com o ensino em programação, sendo dois do gênero masculino e dois do gênero feminino. Três deles possuem doutorado e um possui mestrado, todos na área de Computação. Outro ponto a ressaltar é que os docentes atuam no ensino de disciplinas como Algoritmos e Programação, Estruturas de Dados, Programação Orientada a Objetos, entre outras. Nenhum dos docentes utilizou qualquer tipo de *template* para dar aula utilizando exemplos. Contudo, dois desses docentes (D1 e D2) comentaram que sempre utilizam exemplos em suas aulas, enquanto os outros dois (D3 e D4) utilizam com uma menor frequência.

4.3. Execução

A execução do estudo foi conduzida presencialmente com os docentes. Antes da avaliação, decidimos realizar um estudo piloto afim de verificar se o estudo alcançaria seus objetivos. Os resultados do piloto foram satisfatórios e não foi necessário aprimorar o roteiro do estudo. Cada docente foi convidado por e-mail, que descrevia o objetivo do estudo. Caso aceitassem, um dos pesquisadores agendava uma data para a condução e, em seguida, era enviado um e-mail com o roteiro de preparação do estudo. Nesse documento, estavam disponíveis o formulário *online* do termo de consentimento e o formulário de caracterização. A participação no estudo foi voluntária, e todos os docentes assinaram o termo de consentimento, concordando em participar do estudo e fornecer os resultados para análise. No roteiro, os docentes foram orientados a assistir uma breve apresentação sobre *worked examples* e, em seguida, sobre a proposta do *template*. Foi solicitado também aos docentes que utilizassem e criassem WE a partir do template disponibilizado antes da entrevista.

Na data agendada, conduzimos a avaliação com os docentes, na qual solicitamos que respondessem um questionário com o objetivo de coletar suas experiências e impressões sobre o *template*. O questionário foi elaborado com base nos indicadores TAM [Venkatesh and Davis 2000], que são: **utilidade percebida**, define o grau em que o docente acredita que o uso do **template** pode melhorar seu desempenho no trabalho; **facilidade de uso percebida**, define o grau em que o docente acredita que a adoção do *template* pode ser feita sem muito esforço; e **intenção de uso percebida**: que define o grau em que o docente acredita que poderá utilizar o *template* no futuro. Cada item foi respondido em uma escala de cinco pontos, variando de 'Discordo Fortemente' a 'Concordo Fortemente', com opção neutra. Os itens do questionário podem ser vistos na Tabela 2. Após a a conclusão do questionário, realizamos uma entrevista de acompanhamento, com duração média de 20 minutos, na qual os docentes puderam explicar mais livremente suas percepções sobre o *template*.

Tabela 2. Afirmativas do TAM.

Utilidade Percebida	
UP1	O template de worked examples melhora o meu desempenho contribuindo no planejamento das aulas de programação.
UP2	O template de worked examples melhora a minha produtividade no ensino de programação.
UP3	O template de worked examples facilita o trabalho dos docentes ao ensinar programação aos estudantes.
UP4	Eu acho o template de worked examples útil para auxiliar os docentes no ensino de programação em sala de aula.
UP5	Usando o template de worked examples para apoiar o ensino de programação, eu seria capaz de ensinar os conteúdos mais rapidamente.
Facilidade Percebida	
FUP1	O template de worked examples foi claro e fácil de entender para mim.
FUP2	Usar o template de worked examples não demandou muito esforço mental.
FUP3	Eu acho que o template de worked examples é fácil de usar.
FUP4	Eu acho fácil lembrar como executar as tarefas usando o template de worked examples.
Intenção de Uso	
IU1	Assumindo que eu tenha acesso ao template de worked examples, eu pretendo usá-lo futuramente para me apoiar no ensino de programação.
IU2	Dado que eu tenha acesso ao template de worked examples, eu prevejo que eu o usaria futuramente no ensino de programação.

5. Resultados

Essa seção apresenta os resultados quantitativos e qualitativos obtidos durante a avaliação do *template* pelos participantes.

5.1. Percepções dos docentes sobre o *template*

A Figura 1 ilustra os resultados da análise quantitativa, obtidos através das respostas dos docentes que participaram avaliando o *template* proposto utilizando o método TAM.

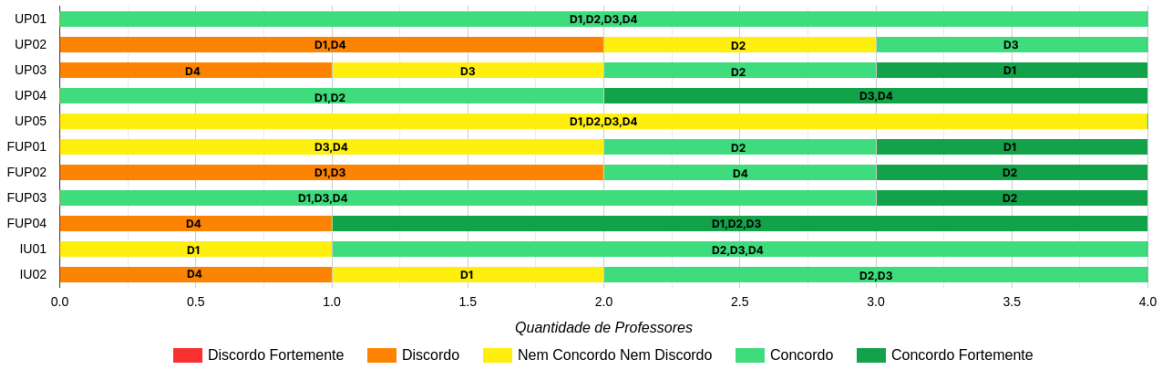

Figura 1. Avaliação do *template* pelos docentes utilizam o TAM

Em relação à **Utilidade Percebida** do *template*, observamos que todos os docentes concordaram sobre a melhora no planejamento das aulas (UP1). No item UP05, todos os docentes permaneceram neutros, indicando que o uso do *template* pode ou não ter impacto na celeridade no processo de ensino do conteúdo. No item UP02, os docentes D1 e D4 não acreditam que o *template* pode causar uma melhoria da produtividade no ensino de programação. Com relação à **Facilidade de Uso Percebida**, observamos divergências de opiniões. No item FUP02, metade dos docentes (D1 e D3) discordaram de que há pouca demanda mental para usar o *template*. Por outro lado, no item FUP04, a maioria dos docentes (três de quatro) considerou fácil memorizar a organização do *template*. Essas divergências podem estar relacionadas à complexidade das disciplinas envolvidas, as quais podem demandar uma carga mental mais alta para uso do *template*.

Quanto à **Intenção de Uso Futuro Percebida**, notamos que houve uma tendência moderada de adesão ao *template* proposto. Em relação ao IU01, três docentes demonstraram interesse em utilizar o artefato em suas práticas acadêmicas. Entretanto, em relação ao IU02, apenas dois docentes concordaram plenamente sobre a previsão de usar o *template*, caso este estivesse disponível. Essa diferença entre as declarações dos participantes e suas intenções reais pode ser atribuída a diversos fatores, como apercepção individual sobre a eficácia e utilidade do *template* dentro de suas próprias competências de ensino.

5.2. Resultados Qualitativos

A partir das entrevistas realizadas, realizamos uma análise qualitativa dos dados e identificamos pontos a partir de três principais categorias.

A primeira categoria está relacionada às **instruções disponíveis**, em que os docentes comentaram sobre a FALTA DE CLAREZA E INSTRUÇÕES DO TEMPLATE. D1 apontou a dificuldade de visualizar a notação a linha a linha no *template*, mencionando que isso seria um problema caso os docentes desejasse fazer uso de fluxogramas: "*pressupõe que estou usando uma notação que vai de linha a linha ... Se fosse um fluxograma, não tem linha, isso poderia ser um problema*". D1 também sugeriu uma melhor estruturação dos testes, tornando-os mais objetivos e claros para os estudantes: *"no teste, eu os esperaria*

em uma perspectiva de Computação, ... com uma entrada e uma saída esperada ... acho que vai ficar mais objetivo.". Complementarmente, D3 destacou a falta de clareza sobre o conceito de teste adotado no *template*: *"pra mim, não ficou claro o que é o teste no template"*. As observações dos docentes evidenciam necessidade de fornecer instruções detalhadas no *template*, explicando o conceito de caso de teste usado e orientando os docentes sobre como inserir os valores indicados para obter a saída esperada.

Notamos ainda que houve uma dificuldade com relação ao LAYOUT E A ORGANIZAÇÃO DOS ELEMENTOS DO TEMPLATE. O docente D4 acredita que é necessário ter uma melhor divisão dos elementos e sugere, *'talvez centralizar, ou destacar mais, ou até dividir"*. Ele complementou dizendo que seria necessário realizar melhorias nos campos de preenchimento, tornando mais claro os pontos a serem preenchidos: "*mais a questão de layout que fiquei dúvida, preencho aqui ou aqui? O visual vai ajudar se ficar mais claro*. Por outro lado, D3 acredita não ser necessário alterar a organização do *template*, pois acredita que " *a estrutura dele [template] está adequada, eu não tiraria nenhuma parte dele*".

Complementarmente, D2 expressou confusão ao não compreender se *" a proposta de solução incorreta é do mesmo algoritmo"*. Esse docente também apontou a redundância do item 'nível de dificuldade' que aparece em dois pontos do *template* e sugeriu que *"poderia estar apenas em cima, assim não repete embaixo"*. D3 sugeriu que talvez *"não precisaria ter nível de dificuldade, pois se eu for aplicar em uma turma, não sei se isso é relevante para aparecer agora, e ainda aparece duas vezes"*. Como pode ser observado, os docentes sugerem a remoção da duplicação do 'nível de dificuldade' e a centralização dele em um único lugar do template, deixando mais claro apenas em relação ao problema. Outro ponto levantado foi a apresentação ou não desse item considerando o contexto de uma turma em que o *template* possa ser aplicado.

Os docentes também forneceram algumas sugestões, que foram agrupadas na categoria **"sugestões de melhoria"**. A primeira sugestão diz respeito à ADEQUAÇÃO DA NOMENCLATURA E TERMINOLOGIA usada no *template*. D1 sugeriu uma definição mais clara para a palavra 'erro', pois a definição dessa palavra pode variar dentro da Computação: *"tinha pensado na nomenclatura que está usada (erro), porque dependendo da área da Computação, a questão da nomenclatura de 'falha', 'erro' ou 'defeito' muda um pouco."* D1 relatou que uma possível melhoria seria fazer CORRELAÇÃO DE DISCIPLINAS E PRÉ-REQUISITOS, uma vez que dependendo do curso de graduação, o nome da disciplina pode ser diferente: *"poderia ter sinônimos dessas disciplinas ou possíveis outros nomes pra elas. Aí você teria tanto conhecimento prévio, como coisas que eles deveriam saber pra fazer esse exemplo, e coisas que eles deveriam saber fazer depois de estudar esse exemplo"*. Dessa forma, ao fornecer essa correlação, o *template* pode se adequar melhor aos diferentes contextos de ensino e garantir que os estudantes tenham a base necessária para aproveitar ao máximo o exemplo proposto.

Uma terceira sugestão está relacionada à CITAÇÃO E AUTORIA DOS WORKED EXAMPLES. D3 acredita ser importante mencionar a autoria: *"fiquei pensando se autoria não seria interessante, acho que uma autoria seria uma diferença aqui [no template]"*. Tal ponto é interessante a ser discutido, até por questão de credibilidade, aumentando assim a confiabilidade do WE disponibilizado aos estudantes. Além disso, D1 sugeriu a adição de CAMPOS OPCIONAIS E INSTRUÇÕES GERAIS: *"faltou adicionar o que é*

opcional e o que não é, talvez, até instruções de como preencher os worked examples em uma visão geral. Os campos opcionais são marcados de algum jeito, ou, então, tá escrito quando é opcional ..., mas preferencialmente coloque os dois". D1 ainda sugeriu a INCLUSÃO DE MÚLTIPLOS TESTES OU CASOS DE TESTE, ou seja, adicionar mais testes, ou um teste que seria opcional para o caso de um erro.

Evidenciando a necessidade de execução dos códigos, D2 sugeriu a adoção de MÚLTIPLAS ABORDAGENS E PROPOSTAS DE SOLUÇÃO. O docente também sugeriu a inclusão de um campo mais amplo para adicionar o código, semelhante a um compilador online, o que permitiria realizar testes rápidos e diretos no código: *"talvez fosse interessante ter um espaço dedicado ao código, parecido com aqueles exemplos que, às vezes, apresentamos no Online GDB (compilador online)"*. D3 endossa, indicando que seria útil uma ferramenta para auxiliar no processo de criação dos WE: *"talvez, seria útil, se houvesse uma ferramenta para me ajudar a preencher esse campo"*. Outro ponto destacado por D4 é que os problemas computacionais podem ter diferentes formas de resolução, portanto, seria interessante possuir mais de um campo para adicionar uma solução diferente: *"existem problemas que podem ter mais de uma solução, talvez, colocar mais de um tipo de solução do problema ... acho que deveria ter opção de criar duas propostas de solução, para o estudante ver a diferença na hora de resolver"*. Com isso em mente, notamos a demanda por uma ferramenta que auxilie na criação de exemplos trabalhados, o que destaca a importância de recursos e suporte para os docentes na elaboração de materiais de ensino.

6. Considerações finais

Fornecer aos docentes uma forma de padronizar e estruturar *worked examples* pode ser essencial para uma melhora considerável nos resultados de aprendizado. Nesse sentido, apresentamos a proposta de um *template* para auxiliar os docentes no uso de WE no ensino de programação. A partir dos resultados do estudo exploratório, percebemos que os docentes tiveram uma receptividade positiva, destacando utilidade do *template* para melhorar o planejamento das aulas e a organização dos WE. Além disso, as discussões dos docentes revelaram *insights* valiosos sobre a aplicabilidade do *template* em diferentes contextos de ensino e as possibilidades de personalização e adaptação para atender às necessidades específicas de cada docente e turma. Entendemos que este é um estudo inicial e que ainda há espaço para aprimoramentos e investigações adicionais. Dado isso, os docentes também forneceram opiniões em relação ao *layout* e a forma como o *template* encontra-se estruturado e concordamos que há a necessidade de realização de melhorias, objetivando uma maior coerência e coesão do *template*.

Foi mencionado também sobre a necessidade de uma solução computacional que simplifique e agilize o uso do *template* no contexto do ensino: aos docentes como forma de criação dos *worked examples* e aos estudantes como forma de apoio a aprendizagem. Nesse sentido, como trabalhos futuros, planejamos integrar o template proposto a um chatbot educacional, que funcionará como uma interface interativa e intuitiva para os docentes. A integração do *template* em um chatbot educacional simplificará o planejamento e a condução das aulas, oferecendo uma experiência mais dinâmica e interativa aos docentes, o que, por sua vez, potencializa o aprendizado dos estudantes e proporciona um ambiente mais estimulante e eficaz para o ensino.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001. Williamson Silva agradece pelo apoio financeiro da FAPERGS (Projeto ARD/ARC – processo 22/2551-0000606-0). Os autores gostariam de agradecer a todos os participantes voluntários do estudo pelo apoio e também ao apoio financeiro Edital nº 122/2022 (PROBIC/FAPERGS).

Referências

Abdul-Rahman, S.-S. and du Boulay, B. (2014). Learning programming via worked-examples: Relation of learning styles to cognitive load. *Computers in Human Behavior*, 30:286–298.

Adams, D. M., McLaren, B. M., Durkin, K., Mayer, R. E., Rittle-Johnson, B., Isotani, S., and Van Velsen, M. (2014). Using erroneous examples to improve mathematics learning with a web-based tutoring system. *Computers in Human Behavior*, 36:401–411.

Atkinson, R. K., Derry, S. J., Renkl, A., and Wortham, D. (2000). Learning from examples: Instructional principles from the worked examples research. *Review of Educational Research*, 70(2):181–214.

Bada, S. O. and Olusegun, S. (2015). Constructivism learning theory: A paradigm for teaching and learning. *Journal of Research & Method in Education*, 5(6):66–70.

Cartwright, N. and McMullin, E. (1984). How the laws of physics lie.

Chen, X., Mitrovic, A., and Mathews, M. (2016). Do erroneous examples improve learning in addition to problem solving and worked examples? In *Intelligent Tutoring Systems: 13th International Conference, ITS 2016, Zagreb, Croatia, June 7-10, 2016. Proceedings 13*, pages 13–22. Springer.

Garces, S., Vieira, C., Ravai, G., and Magana, A. J. (2023). Engaging students in active exploration of programming worked examples. *Education and Information Technologies*, 28(3):2869–2886.

Große, C. S. and Renkl, A. (2007). Finding and fixing errors in worked examples: Can this foster learning outcomes? *Learning and instruction*, 17(6):612–634.

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2):4.

Hosseini, R., Akhuseyinoglu, K., Brusilovsky, P., Malmi, L., Pollari-Malmi, K., Schunn, C., and Sirkiä, T. (2020). Improving engagement in program construction examples for learning python programming. *International Journal of Artificial Intelligence in Education*, 30(2):299–336.

Huang, Y.-H., Lin, K.-C., Yu, X., and Hung, J. (2015). Comparison of different approaches on example-based learning for novice and proficient learners. *Human-centric Computing and Information Sciences*, 5.

Kopp, V., Stark, R., and Fischer, M. R. (2008). Fostering diagnostic knowledge through computer-supported, case-based worked examples: effects of erroneous examples and feedback. *Medical education*, 42(8):823–829.

McGinn, K. M., Lange, K. E., and Booth, J. L. (2015). A worked example for creating worked examples. *Mathematics Teaching in the Middle School*, 21(1):26–33.

McLaren, B. M., van Gog, T., Ganoe, C., Karabinos, M., and Yaron, D. (2016). The efficiency of worked examples compared to erroneous examples, tutored problem solving, and problem solving in computer-based learning environments. *Computers in Human Behavior*, 55:87–99.

Monk, P. M. (2008). *Fundamentals of electroanalytical chemistry*. John Wiley & Sons.

Renkl, A. (2014). The worked examples principle in multimedia learning.

Renkl, A. (2017). Learning from worked-examples in mathematics: Students relate procedures to principles. *ZDM*, 49(4):571–584.

Retnowati, E. and Marissa (2018). Designing worked examples for learning tangent lines to circles. *Journal of Physics: Conference Series*, 983:012124.

Rittle-Johnson, B., Loehr, A. M., and Durkin, K. (2017). Promoting self-explanation to improve mathematics learning: A meta-analysis and instructional design principles. *ZDM*, 49:599–611.

Rodiawati, A. and Retnowati, E. (2019). How to design worked examples for learning patterns in mathematics. *Journal of Physics: Conference Series*, 1320:012045.

Sweller, J., Van Merrienboer, J. J., and Paas, F. G. (1998). Cognitive architecture and instructional design. *Educational psychology review*, pages 251–296.

Tonhão, S., Colanzi, T., and Steinmacher, I. (2021). Using real worked examples to aid software engineering teaching. In *Proceedings of the XXXV Brazilian Symposium on Software Engineering*, SBES '21, page 133–142, New York, NY, USA. Association for Computing Machinery.

van Merrienboer, J. J. (1990). Instructional strategies for teaching computer programming: Interactions with the cognitive style reflection-impulsivity. *Journal of research on computing in education*, 23(1):45–53.

Venkatesh, V. and Davis, F. D. (2000). A theoretical extension of the technology acceptance model: Four longitudinal field studies. *Management science*, 46(2):186–204.