Utilizando Aprendizado Baseado em Problemas para o Ensino de Paradigmas de Programação

Alice Fonseca Finger[1], **João Pablo Silva da Silva**[1], **Miguel Ecar**[1]

[1]LabISE - Laboratory of Intelligent Software Engineering
Universidade Federal do Pampa (UNIPAMPA)
Av. Tiarajú, 810, Ibirapuitã – Alegrete, RS – Brasil

{alicefinger, joaosilva}@unipampa.edu.br; miguel@ecarsm.com

Abstract. *In this work, we present the results of software development paradigms teaching-learning using Problem-Based Learning (PBL). For this, we describe our methodology in the form of a teaching plan and report the obtained execution results in a class during the first academic semester of 2020. As a result, we observed a good final performance in the class, highlighting a higher ease in dealing with the procedural paradigm and a high difficulty in dealing with the logical and functional paradigms. The main PBL using contribution is the autonomy developed by the students, mainly in the paradigms considered more difficult.*

Resumo. *Neste trabalho apresentamos os resultados da aplicação da metodologia de Aprendizado Baseado em Problemas (ABP) no ensino-aprendizagem de paradigmas de programação. Para isso, descrevemos a nossa metodologia na forma de um plano de ensino e reportamos os resultados obtidos com a sua execução em uma turma no primeiro semestre letivo de 2020. Como resultado, obtivemos um bom desempenho final da turma, destacando uma maior facilidade em lidar com o paradigma procedimental e uma maior dificuldade em lidar com os paradigmas lógico e funcional. A principal contribuição do uso de ABP está na autonomia desenvolvida pelos estudantes, principalmente nos paradigmas considerados mais difíceis.*

1. Introdução

O Curso de Engenharia de Software do Campus Alegrete da Universidade Federal do Pampa tem por objetivo promover ensino, pesquisa e extensão em Engenharia de Software (ES), contribuindo com o desenvolvimento sustentável da região e do país. O Curso tem uma carga horária de 3300 horas distribuídas ao longo de 9 semestres, cobrindo conteúdos dos seguintes eixos de conhecimento: fundamentos da matemática, fundamentos da computação, engenharia de software e contexto profissional. Além dos aspectos técnicos, o Curso busca desenvolver também as competências e habilidades relacionadas à gestão, consultoria, pesquisa e empreendedorismo [UNIPAMPA 2018].

Desde sua criação, o Curso adota a ABP, a qual desenvolve as competências e habilidades a partir de problemas reais, melhorando a relação entre teoria e prática e tornando os estudantes protagonistas no processo de aprendizagem [UNIPAMPA 2018]. ABP é uma abordagem de ensino-aprendizagem construtivista que explora diversos tipos de problemas, permitindo o desenvolvimento do raciocínio lógico, da criatividade e da

DOI: 10.5753/sbie.2021.218493

interpretação de textos, além de aumentar a motivação dos estudantes quando da resolução do problema [Martins 2002].

ABP está implementada no currículo do Curso na forma de um conjunto de seis disciplinas. Essas disciplinas integram, de modo interdisciplinar e transversal, diferentes conteúdos na abordagem de uma situação-problema que se aproxima da realidade profissional que os egressos poderão encontrar. Nas disciplinas os estudantes são organizados em times com o intuito de desenvolver a habilidade de trabalhar de forma colaborativa. O processo de avaliação é contínuo e acumulativo, observando as dimensões individuais e coletivas. A dimensão individual avalia o empenho do estudante na busca pela solução, enquanto que, a dimensão coletiva avalia o resultado produzido pelo time como solução [UNIPAMPA 2018].

Em 2018, motivado pela necessidade de adequar-se às Diretrizes Curriculares Nacionais (DCNs) para os cursos de graduação na área da computação, o Curso implantou seu novo Projeto Pedagógico de Curso (PPC). Entre as novidades, destaca-se a disciplina Resolução de Problemas III (RP III), a qual usa ABP para desenvolver as competências e habilidades relacionadas aos paradigmas de programação. Essa disciplina é ofertada para as turmas do terceiro semestre do Curso, possuindo uma carga horária de 120 horas, das quais 60 horas são presenciais e 60 horas são não presenciais. Até o momento, foram executadas duas edições, sendo nos primeiros semestres de 2019 e 2020.

Nosso objetivo é apresentar os resultados da aplicação de uma metodologia ativa no ensino de paradigmas de programação. Para tanto, primeiramente apresentamos a estrutura metodológica da disciplina RP III na forma de um plano de ensino. Após, relatamos a execução do plano de ensino proposto, ilustrando-o com dados quantitativos (desempenho dos estudantes) e qualitativos (percepção dos estudantes). Por fim, é feita uma discussão dos resultados a fim de propor melhorias nas próximas execuções da disciplina, bem como avaliar os pontos fortes dessa edição. Cabe observar que os resultados apresentados aqui se referem a execução de RP III no primeiro semestre letivo de 2020.

O restante deste trabalho está organizado como segue. Na Seção 2 reportamos alguns trabalhos relacionados ao uso de ABP para ensino de ES. Na Seção 3 descrevemos o planejamento da disciplina RP III, o qual engloba o conteúdo de paradigmas de programação. Na Seção 4, detalhamos os resultados quantitativos e qualitativos do desempenho dos estudantes em cada paradigma, bem como a percepção deles quanto ao componente. Na Seção 5 apresentamos as considerações finais e os trabalhos futuros.

2. Trabalhos Relacionados

Analisando alguns estudos encontrados na literatura especializada através de uma busca *ad hoc*, percebemos duas abordagens distintas para o uso de ABP.

Na primeira abordagem, ABP é usada como eixo metodológico dos cursos, influenciando a execução de um conjunto de disciplinas e estando ligada diretamente ao perfil dos estudantes que se deseja formar. Nesse sentido, destacamos o estudo de Santos *et al.* [Santos et al. 2008], no qual é relatado o uso de ABP em um curso de mestrado profissional para desenvolver as capacidades de lidar com problemas reais. Também destacamos os estudos de Cera *et al.* [Cera et al. 2012] e Guedes *et al.* [Guedes et al. 2017], nos quais são reportados estudos de caso sobre o uso de ABP como base metodológica de cursos de graduação para obter uma melhor relação entre teoria e prática.

Na segunda abordagem, ABP é usada isoladamente com o objetivo de desenvolver competências e habilidades específicas dentro de um eixo de formação. Nesse sentido, destacamos o estudo de Richardson e Delaney [Richardson and Delaney 2009], no qual é reportado um estudo de caso sobre a aplicação de ABP em uma disciplina de engenharia de software para o ensino de técnicas de diagramação. Também destacamos os estudos de Figuerêdo *et al.* [Figuerêdo, C.O.; Santos, S.C., Borba, P.H.; Alexandre, G.H. 2011], Richardson *et al.* [Richardson et al. 2011], Hoffmann *et al.* e Cheiran *et al.* [Cheiran et al. 2017], nos quais são apresentados estudos de caso sobre o uso de ABP no desenvolvimento das competências e habilidades para engenheiros de testes.

Neste trabalho, exploramos características de ambas abordagens. Usamos ABP para desenvolver competências e habilidades específicas, no caso paradigmas de programação, guiada por um eixo metodológico que orienta o uso de metodologias ativas durante o percurso formativo dos estudantes. Da mesma forma que nos trabalhos citados, temos percebido os benefícios não só nas capacidades técnicas, mas principalmente nas competências e habilidades comportamentais dos egressos.

3. Planejamento de Ensino

A disciplina RP III busca tornar os estudantes aptos a resolver problemas através do desenvolvimento de software com diferentes paradigmas de programação. Complementarmente, espera-se que os estudantes sejam capazes de: i) abstrair as principais características dos principais paradigmas de programação; ii) escolher a linguagem adequada, levando em consideração aspectos relevantes ao problema; e iii) programar e testar softwares desenvolvidos com diferentes paradigmas de programação [UNIPAMPA 2018].

A ementa é composta por programação procedimental, programação lógica, programação funcional e programação orientada a aspectos. Destacamos que até o terceiro semestre do Curso, os estudantes têm contato com paradigma orientado a objetos e 15 horas práticas de Prolog dentro da disciplina de Lógica Matemática. Conforme PPC do Curso, RP III é sempre ministrada por um par de docentes. Os docentes são responsáveis pelo planejamento e execução da disciplina. No contexto de ABP, os docentes atuam como facilitadores do processo de ensino-aprendizagem, provendo suporte e orientação para os estudantes [UNIPAMPA 2018].

RP III possui uma carga horária de 120 horas, das quais 60 horas são presenciais e 60 horas são não presenciais. Devido à pandemia de Covid-19, o primeiro semestre letivo de 2020 foi executado totalmente de forma remota [UNIPAMPA 2020]. Por esse motivo, a RP III reportada neste trabalho foi planejada para ser ministrada de forma remota em uma sala virtual da plataforma Google Meet, ocorrendo de maneira **síncrona**, desenvolvida em tempo real pelo docente com a participação simultânea dos estudantes; e de maneira **assíncrona**, realizada em tempos diversos, não exigindo a participação simultânea no mesmo espaço e tempo. Para disponibilização de material e controle das atividades criamos um curso na plataforma Moodle da instituição, específico para esta edição da RP III. Complementarmente, criamos repositórios no servidor Git[1] institucional para monitorar e controlar os códigos-fonte produzidos pelos discentes.

No que tange ABP, na RP III os estudantes são desafiados a resolver problemas usando diferentes paradigmas de programação. Cabe observar que os estudantes são os

[1]Para mais informações sobre a ferramenta Git acesse `https://git-scm.com/`.

protagonistas do processo de ensino-aprendizagem, ou seja, eles são responsáveis por desenvolver as competências e habilidades necessárias para resolver o problema. Para isso, a turma deve auto-organizar-se em grupos, os quais trabalham para resolver os problemas de forma autodidata, pró-ativa e colaborativa.

Para esta edição da RP III planejamos 4 iterações, onde cada iteração teve por meta desenvolver as competências e habilidades de um paradigma de programação. No início de cada iteração, os docentes devem apresentar o problema e o paradigma a ser usado para resolvê-lo. Ao final de cada iteração, cada grupo deve entregar e apresentar um produto de software desenvolvido como solução para o problema. As iterações foram organizadas assim:

- **Iteração 1** – Paradigma Procedimental – 3 semanas de duração - Desenvolver um software para gerenciar o acervo de filmes;
- **Iteração 2** – Paradigma Lógico – 4 semanas de duração – Desenvolver um software para criar versões de horários de aula;
- **Iteração 3** – Paradigma Funcional – 4 semanas de duração – Desenvolver um software para gerenciar uma locadora de carros;
- **Iteração 4** – Paradigma Orientado a Aspectos – 3 semanas de duração – Desenvolver um software para gerenciar uma loja de artigos esportivos.

Conforme PPC do Curso, as avaliações da RP III devem ser feitas sob duas perspectivas. Na perspectiva individual deve ser avaliado o comprometimento e o desempenho do discente no processo de desenvolvimento da solução. Na perspectiva em grupo deve ser avaliado a solução produzida pelo grupo [UNIPAMPA 2018]. Nesta edição da RP III cada perspectiva teve um peso cinco na composição da nota final.

Na perspectiva individual, semanalmente, cada discente deve reportar e evidenciar o andamento de seu trabalho. Os reportes são feitos em tarefas específicas criadas no Moodle, onde cada estudante relata o trabalho feito e a referencia, informando identificadores dos *commits*[2] feitos no repositório Git. Os docentes realizam uma breve arguição sobre o trabalho para se certificarem que o discente tinha propriedade sobre o que está sendo mostrado. A partir dessa arguição, deve ser atribuído um dos seguintes conceitos: Não Ok (NOk), quando não há trabalho ou o trabalho é insatisfatório; Parcialmente Ok (POk), quando há trabalho em pouco volume ou qualidade; Totalmente Ok (TOk), quando o trabalho está adequado. Nesta edição, planejamos dez verificações semanais, as quais também são usadas para sanar dúvidas e prover orientações para cada grupo de trabalho.

Na perspectiva do grupo, ao final de cada iteração, os grupos apresentam seus produtos. Os docentes também realizam uma breve arguição sobre o produto entregue/apresentado para se certificarem que o grupo tem domínio do produto. Após a arguição, os docentes analisam o produto quanto à correção e suficiência, além de aspectos técnicos. Cada grupo deve receber uma nota entre zero e dez. Nesta edição, planejamos quatro entregas de iteração, as quais também são usadas para fazer uma reflexão sobre as lições aprendidas pelo grupo, principalmente no que tange à autogestão.

4. Resultados Obtidos

A edição da RP III do primeiro semestre letivo de 2020 contou com um total de 36 discentes matriculados. A turma foi organizada em 9 grupos de trabalho, sendo 7 compostos

[2] *Commit* é o comando usado para submeter modificações de artefatos para o repositório.

por 4 integrantes e 2 compostos por 3 integrantes. Como resultado final, obtivemos 26 aprovações, 6 reprovações por nota, 2 reprovações por frequência e 2 trancamentos de matrícula.

4.1. Desempenho Individual

A primeira iteração objetivou desenvolver as competência e habilidades relacionadas ao paradigma de programação procedimental. Para tanto, cada grupo teve que desenvolver um software de gerenciamento de locadora de filmes em linguagem de programação C. A Figura 1 mostra o desempenho dos estudantes nesta iteração.

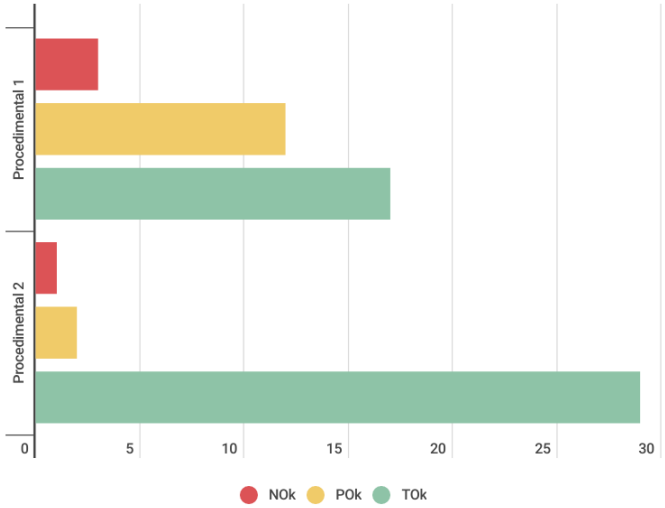

Figura 1. Conceitos das verificações do paradigma procedimental.

A partir do gráfico da Figura 1, podemos perceber uma evolução da verificação 1 para a 2, já que a quantidade de conceitos TOk aumentou consideravelmente e, da mesma maneira NOk e POk diminuíram. Esse desempenho condiz com o bom desenvolvimento que os estudantes mostraram para a programação procedimental.

A segunda iteração buscou desenvolver as competências e habilidades relacionadas ao paradigma de programação lógica. Para isso, cada grupo teve que desenvolver um software que criasse versões de horários de aula para o primeiro e terceiro semestres de um curso de graduação, a linguagem de programação usada nesta iteração foi Prolog. A Figura 2 apresenta o desempenho dos estudantes nesta iteração.

Notamos que os estudantes não tiveram um desempenho regular durante as verificações desta iteração. Na primeira verificação semanal obtivemos um desempenho bom, dada a quantidade de conceito TOk. Porém, nas duas semanas seguintes esse rendimento caiu consideravelmente, aumentando os valores para NOk e POk. Para a grande maioria dos estudantes, esse foi um dos paradigmas mais difíceis. A justificativa para a primeira semana ter um rendimento bom se dá porque naquela verificação apenas fatos foram criados no Prolog, o que de certa maneira era considerado mais fácil dentro do escopo de desenvolvimento.

A terceira iteração buscou desenvolver as competências e habilidades relacionadas ao paradigma de programação funcional. Para tanto, cada grupo teve que desenvolver um

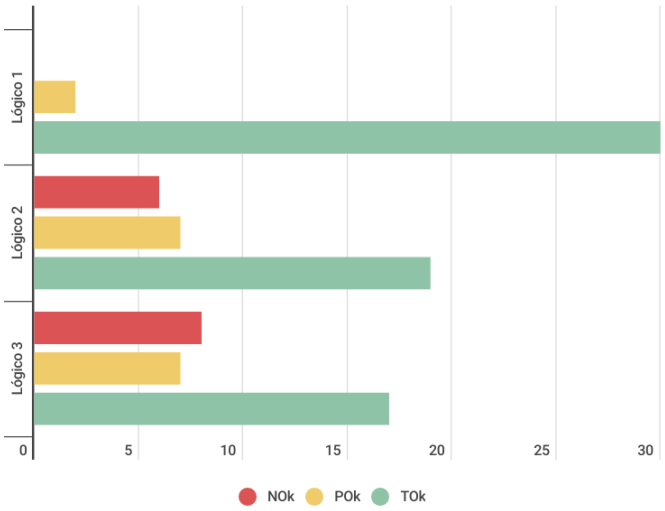

Figura 2. Conceitos das verificações do paradigma lógico.

software para o gerenciamento de uma locadora de carros em linguagem de programação Haskell. A Figura 3 apresenta o desempenho dos estudantes nesta iteração.

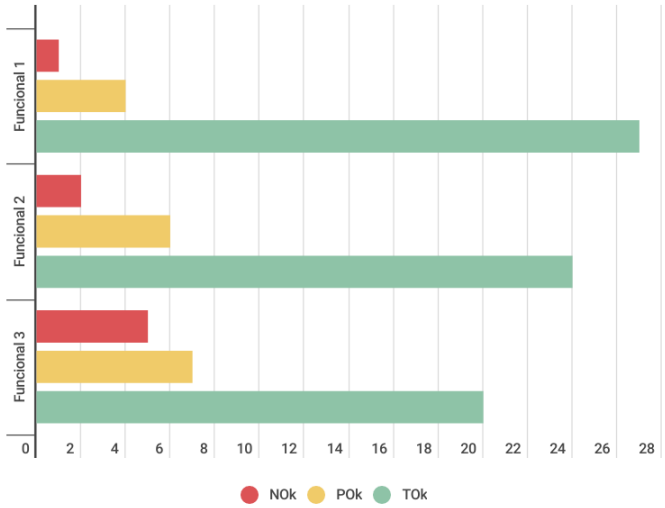

Figura 3. Conceitos das verificações do paradigma funcional.

Observamos que o desempenho dos estudantes nas verificações semanais foi decrescendo, começando com uma boa quantidade de conceito TOk e diminuindo ao longo das outras duas verificações. A justificativa para esse desempenho se dá pelo fato de que o paradigma funcional exige um conhecimento maior em conceitos matemáticos e, a medida em que passam as semanas, o escopo do desenvolvimento aumenta de complexidade, exigindo cada vez mais conhecimentos dos estudantes.

A quarta iteração buscou desenvolver as competências e habilidades relacionadas ao paradigma de programação orientado a aspectos. Para isso, cada grupo teve que desenvolver um software para o gerenciamento de uma loja de artigos esportivos em AspectJ, uma extensão da linguagem de programação Java orientada à aspectos de propósito geral. A Figura 4 apresenta o desempenho dos estudantes nesta iteração.

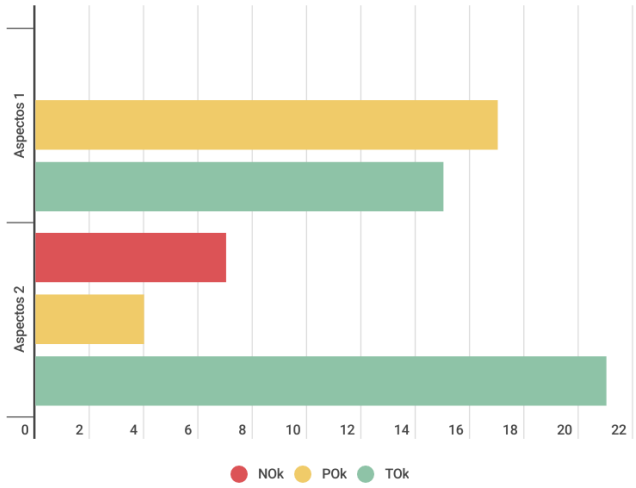

Figura 4. Conceitos das verificações do paradigma orientado a aspectos.

Percebemos um desempenho regular dos estudantes nas verificações semanais do paradigma orientado a aspectos. Acreditamos que o aumento de conceitos TOk se deu pela familiaridade que os estudantes já tinham com a o paradigma orientado a objetos e linguagem de programação Java.

4.2. Desempenho em Grupo

Ao final de cada iteração, os grupos realizaram uma apresentação do produto desenvolvido, e, após arguição e análise do trabalho desenvolvido, os docentes avaliaram com uma nota de 0 à 10. Abaixo, a Figura 5 apresenta um boxplot das notas dos grupos em cada paradigma.

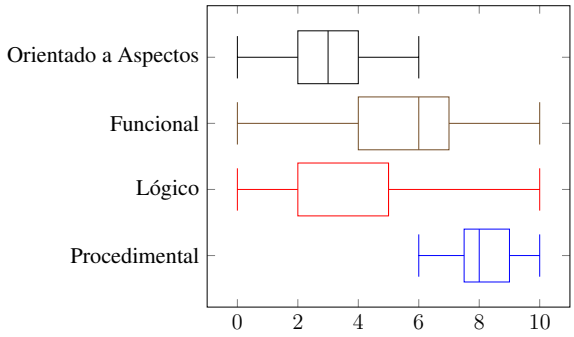

Figura 5. Boxplot das notas em grupo para cada paradigma.

Analisando o desempenho geral, é possível perceber que o paradigma procedimental, assim como nas verificações individuais, apresentou um melhor desempenho dos grupos, com nota mínima de 6 e máxima de 10 no paradigma. Nos demais, fica visível que pelo menos um grupo em cada paradigma obteve nota 0. Chamamos atenção para o paradigma orientado à aspectos, no qual a nota máxima dos grupos não passou de 6. Acreditamos que isso se deu devido à alguns grupos já estarem com nota suficiente para aprovação e, por isso, não se empenharam em desenvolver o produto.

4.3. Percepção dos Estudantes

Ao final desta edição, aplicamos um questionário para avaliarmos a absorção dos conhecimentos sobre paradigmas de programação pelos estudantes e a percepção de cada um sobre a condução da disciplina. O questionário, respondido voluntariamente, era composto por 2 questões sobre o conceito de paradigmas de programação e 4 questões sobre a opinião dos estudantes em relação à disciplina. Do total de 36 estudantes matriculados, obtivemos 14 respostas.

A questão de conceito geral sobre paradigmas de programação teve 100% de acerto. Já a segunda questão, referente a um esquema de relacionar o paradigma com a sua definição, apresentou diferentes respostas, como mostra a Figura 6.

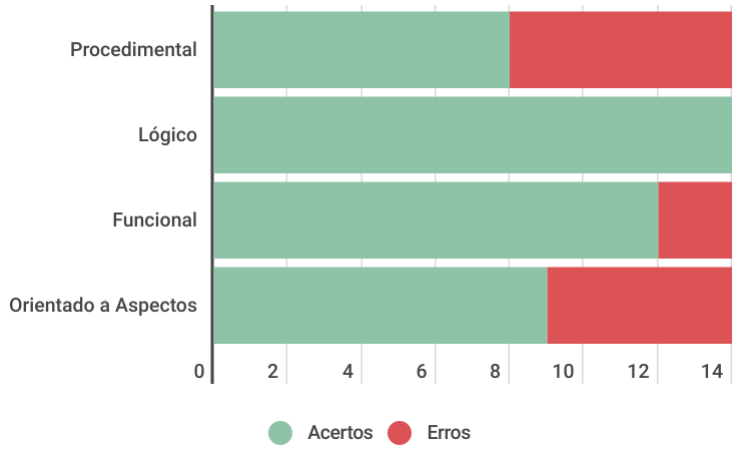

Figura 6. Questão de relacionar o paradigma com sua definição.

A partir das respostas dos estudantes, notamos que, mesmo com a dificuldade no desenvolvimento usando o paradigma lógico, o entendimento de suas particularidades foi aprendido. Por outro lado, mesmo com a facilidade que os estudantes demonstraram na linguagem de programação C, a característica principal de modularização do código para melhor manutenibilidade não foi entendido por completo no paradigma procedimental.

A Figura 7 apresenta a opinião dos estudantes em relação às seguintes perguntas: Qual paradigma você classifica como mais fácil? Qual paradigma você classifica como mais difícil?

Ao analisar o resultado, percebemos que o paradigma procedimental foi considerado, pela maioria, o mais fácil, já o paradigma lógico o mais difícil. Esse resultado já era esperado, visto o desempenho apresentado pelos estudantes nos dois paradigmas.

Com a finalidade de uma avaliação qualitativa da disciplina, o questionário continha a seguinte questão: Na sua opinião, desenvolver em uma linguagem de programação específica é suficiente para aprender o paradigma de programação que está sendo utilizado? Comente.

A partir de uma análise de conteúdo, chegou-se a seguinte síntese de respostas: 5 disseram que sim, é suficiente; 7 responderam que não é suficiente; 2 falaram que talvez seja suficiente.

Observando os comentários em relação a resposta “Não”, alguns chamam a

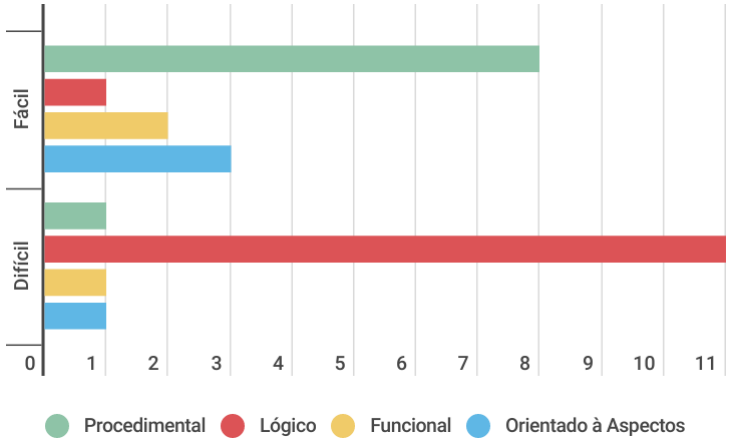

Figura 7. Paradigmas mais fáceis e difíceis na visão dos estudantes.

atenção: "Não, talvez se fosse possível escolher uma linguagem de tal paradigma, sem ser uma já determinada, pudesse ser mais interessante.", "Não, muitas vezes só pegar e sair escrevendo não é o suficiente pois um novo paradigma é algo totalmente diferente de uma nova linguagem, é preciso aprender conceitos nunca antes vistos e mudar o modo com que pensamos e tomamos decisões. Isso leva tempo e muitas vezes o programador fica perdido pois só conhece/foi exposto ao procedimental."

Já em relação as respostas "Sim", destaca-se: "Acho que sim, o paradigma não depende da linguagem utilizada mas sim do empenho do estudante a se dedicar a teoria deste paradigma." Por fim, para aqueles que responderam "Talvez", destaca-se: "Acredito que sirva de base, mas dificilmente vamos aprender o paradigma completo.", "Mais ou menos, na prática se aprende muita coisa, porém sem uma base teórica a prática praticamente não existe."

5. Considerações Finais

Apresentamos neste trabalho a aplicação da metodologia ABP no ensino-aprendizagem de paradigmas de programação. Como resultado final, a grande maioria dos estudantes conseguiu a aprovação na disciplina. Analisando os resultados parciais, percebemos uma maior facilidade com o paradigma procedimental e uma maior dificuldade com os paradigmas lógico e funcional. No que se refere ao uso de ABP, a principal contribuição percebida foi em relação à autonomia dos estudantes. Essa percepção se fortaleceu justamente nos paradigmas considerados mais difíceis, já que eles partiram de uma base bem elementar e desenvolveram as competências e habilidade necessárias para aprovação de forma autônoma e colaborativa. Cabe observar que essa percepção se aplica ao contexto deste trabalho, não podendo ser generalizado para outros.

Uma limitação percebida em nossa metodologia está ligada à necessidade de tornar mais forte a relação entre teoria e prática. Percebemos que, para alguns estudantes, mesmo conseguindo resolver o problema, alguns conceitos essenciais não foram bem fixados. Entendemos que este é um efeito colateral que obtivemos ao dar foco na aplicação prática dos conteúdos relacionados na ementa. Como trabalho futuro, pretendemos integrar à nossa metodologia baseada em ABP técnicas de gamificação [Ecar and da Silva 2020], no intuito de equilibrar a ênfase entre teoria e prática e conti-

nuar o ensino com uso de metodologias ativas para um melhor envolvimento dos alunos.

Referências

Cera, M., Forno, M. D., and Vieira, V. G. (2012). A proposal to teach software engineering based on problem solving. *Brazilian Journal of Computers in Education*, 20(3):116–132.

Cheiran, J. F. P., Rodrigues, E. d. M., Carvalho, E. L. S., and da Silva, J. P. S. (2017). Problem-Based Learning to Align Theory and Practice in Software Testing Teaching. In *Proceedings of the 31st Brazilian Symposium on Software Engineering*, pages 328–337, New York. ACM Press.

Ecar, M. and da Silva, J. P. (2020). Assessment gamification with fhoment methodology: An action research based case study. In *Anais do XXXI Simpósio Brasileiro de Informática na Educação*, pages 582–591, Porto Alegre, RS, Brasil. SBC.

Figuerêdo, C.O.; Santos, S.C., Borba, P.H.; Alexandre, G.H. (2011). Using pbl to develop software test engineers. In *14th IASTED International Conference on Computers and Advanced Technology in Education*, page 7, Cambridge. Acta.

Guedes, G. T. A., Bordin, A. S., Mello, A., and Melo, A. M. (2017). Pbl integration into a software engineering undergraduate degree program curriculum: An analysis of the students' perceptions. In *Proceedings of the 31st Brazilian Symposium on Software Engineering*, pages 308–317.

Martins, J. G. (2002). *Aprendizagem Baseada em Problemas Aplicada a Ambiente Virtual de Aprendizagem*. PhD thesis, Universidade Federal de Santa Catarina.

Richardson, I. and Delaney, Y. (2009). Problem based learning in the software engineering classroom. In *22nd Conference on Software Engineering Education and Training*, pages 174–181, Washington. IEEE.

Richardson, I., Reid, L., Seidman, S. B., Pattinson, B., and Delaney, Y. (2011). Educating software engineers of the future: Software quality research through problem-based learning. In *24th IEEE-CS Conference on Software Engineering Education and Training*, pages 91–100, Honolulu. IEEE.

Santos, S. C., Batista, M. C. M., Cavalcanti, A. P., Albuquerque, J. O., and Meira, S. (2008). Usando pbl na qualificação de profissionais em engenharia de software. In *I Fórum de Educação em Engenharia de Software*, page 9, Campinas. ACM.

UNIPAMPA (2018). Projeto Pedagógico do Curso de Engenharia de Software. Disponível em: `http://dspace.unipampa.edu.br/bitstream/riu/100/11/PPC\%20Engenharia\%20de\%20Software\_Alegrete`, Acessado em: 29/06/21.

UNIPAMPA (2020). Diretrizes Operacionais Para Oferta das Atividades de Ensino Remoto Emergenciais. Disponível em: `https://sites.unipampa.edu.br/prograd/files/2020/08/norma-operacinal-n-o-4-2020_diretrizes-operacionais-para-oferta-das-atividades-de-ensino-remoto-emergenciais.pdf`, Acessado em: 30/06/21.