Hall of Fame/Shame: um Padrão Pedagógico para o Ensino de Programação

Andrea S. Charão[1,2], Alberto F. Kummer Neto[2]
Benhur de O. Stein[1,2], Patrícia P. de A. Barcelos[1]

[1] Departamento de Linguagens e Sistemas de Computação

[2]Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria
Santa Maria, RS, Brasil

{andrea, alberto, benhur, pitthan}@inf.ufsm.br

Resumo. *Padrões pedagógicos visam aproveitar o conhecimento especializado em práticas de ensino e aprendizagem, em um formato que favoreça sua reutilização. Neste trabalho, propõe-se um padrão pedagógico voltado ao ensino de programação, em cursos de nível superior na área de Computação. Este padrão é centrado na exposição e discussão de bons e maus exemplos de códigos produzidos pelos alunos, formando o que denominou-se respectivamente de "Hall of Fame" e "Hall of Shame". Ao longo do texto, apresenta-se uma caracterização do padrão, relacionando-o com outros padrões pedagógicos descritos na literatura, e descreve-se sua aplicação no ensino de programação orientada a objetos, em uma instituição de ensino superior. Os resultados indicam uma avaliação positiva do padrão pelos alunos e revelam sua eficácia em incentivar boas práticas de programação.*

Abstract. *Pedagogical patterns propose to take advantage of the expertise in teaching and learning practices, in an organized manner that can be easily reused. In this paper, we propose a pedagogical pattern targeted to teaching programming in higher level courses in Computer Science. This pattern is centered on the presentation and discussion of good and bad examples of codes produced by the students, forming what was named respectively "Hall of Fame" and "Hall of Shame". Throughout the text, we presents a characterization of HoFS pattern, relating it to other pedagogical patterns described in the literature. We also report its application in teaching object oriented programming in a higher education institution. The results indicate a positive evaluation by students and reveal the pattern helps to encourage good programming practices.*

1. Introdução

Os processos de ensino-aprendizagem de programação de computadores são alvo de discussões recorrentes em comunidades de pesquisadores e professores da área de Computação [Pears et al. 2007, Aureliano e Tedesco 2012]. É consenso que a programação impõe várias dificuldades a alunos iniciantes e, aos que já possuem alguma experiência prévia, apresenta-se também o desafio de aprimorar suas habilidades e aprender novos paradigmas [Jenkins 2002, Robins et al. 2003, Lahtinen et al. 2005, Tan et al. 2009]. Aos professores e instituições de ensino, cabe a responsabilidade de permanentemente refletir

sobre as metodologias e as práticas de ensino-aprendizagem de programação, que envolve também a tomada de decisões com impacto em todo o processo [Denning 1989]. Questões frequentes neste contexto incluem, por exemplo: Que linguagens de programação adotar para iniciantes? Que estratégias de ensino-aprendizagem são mais eficazes?

Em buscas de respostas para estas e outras questões, há exemplos de pesquisadores que têm se dedicado a compreender melhor os processos cognitivos envolvidos na aprendizagem de programação [Soloway e Spohrer 1988, Winslow 1996], enquanto outros investigam e propõem formas de lidar com as dificuldades recorrentes [Wilson e Shrock 2001, Caspersen e Kolling 2009]. Nessa segunda direção, há diversos autores que sustentam uma abordagem centrada nos chamados "padrões pedagógicos" (*pedagogical patterns*) [Sharp et al. 1996], originários de uma comunidade interessada no ensino-aprendizagem de programação orientada a objetos, e gradativamente estendidos para um escopo mais amplo [Larson et al. 2008, Köppe 2015]. Essa abordagem propõe-se a capturar boas práticas de ensino-aprendizagem em domínios específicos, sob uma forma compacta e organizada, para que possa ser facilmente comunicada e reutilizada. Seu foco principal é, portanto, facilitar a disseminação de boas práticas.

Desde a proposta inicial, em 1996, vários foram os padrões pedagógicos descritos, catalogados e publicados, com diferentes intenções. No domínio específico de aprendizagem de programação, há padrões variados, como por exemplo: *Mistake* [Bergin 2000], que propõe que os alunos criem programas (ou outros artefatos) com erros; e *Show Programming*, que sugere mostrar programação em tempo real aos alunos, e não somente slides sobre programação [Schmolitzky 2007]. Conforme autores da área [Bergin et al. 2012], alguns padrões podem parecer triviais para educadores experientes, mas se tornam referências úteis para quem tem menos experiência ou busca diversificar suas práticas. A forma sucinta de apresentação dos padrões facilita este compartilhamento; por outro lado, a ausência de dados complementares sobre avaliação de certos padrões pode suscitar dúvidas quanto à sua efetividade.

Neste artigo, apresenta-se um padrão pedagógico voltado ao ensino de programação, em cursos de nível superior na área de Computação. Este padrão, que denominou-se "Hall of Fame/Shame" (HoFS), é centrado na exposição e discussão de bons e maus exemplos de códigos produzidos pelos alunos. Sua motivação é principalmente estimular os alunos a aprimorarem suas habilidades em programação. Além de descrever o padrão de forma sucinta, num formato que facilite o reuso, apresenta-se também resultados experimentais sobre sua aplicação prática numa instituição de ensino superior.

Este artigo encontra-se assim organizado: na seção 2 apresenta-se sucintamente uma visão histórica de padrões pedagógicos no ensino de programação, indicando as principais fontes de referência sobre o assunto, até os dias atuais; na seção 3 apresenta-se o padrão HoFS num formato tipicamente utilizado pela comunidade da área; na seção 4 descreve-se a aplicação e a avaliação do padrão em uma instituição de ensino superior; na seção 5, por fim, apresentam-se considerações finais sobre o trabalho.

2. Padrões Pedagógicos no Ensino de Programação

Padrões pedagógicos (*pedagogical patterns*) [Sharp et al. 1996] surgiram no contexto de discussões sobre ensino de programação orientada a objetos e suas tecnologias. A inspiração para isso veio dos chamados padrões de projeto (*design patterns*) [Gamma et al. 1993],

que constituem soluções bem sucedidas e reusáveis para problemas no contexto da orientação a objetos. A ideia inicial dos padrões pedagógicos era, portanto, usar essa mesma abordagem para catalogar soluções reusáveis a problemas comumente encontrados no ensino desse paradigma.

Gradativamente, a ideia dos padrões pedagógicos estendeu-se a um escopo mais amplo, capturando soluções para problemas encontrados em diferentes situações e domínios além da orientação a objetos. Pesquisadores e educadores passaram a propor padrões em conferências tais como PLoP (Pattern Languages of Programs) e EuroPLoP (European Conference on Pattern Languages of Programs), chegando a conferências tradicionais sobre ensino de computação, tais como SIGCSE (ACM Technical Symposium on Computing Science Education) e ITiCSE (ACM Conference on Innovation and Technology in Computer Science Education). Os padrões foram sendo catalogados em sites[1][2] e, mais recentemente, vários deles forem reunidos em livros [Bergin et al. 2012, Mor et al. 2014].

No cenário brasileiro, um dos primeiros trabalhos que encontramos sobre padrões pedagógicos foi uma dissertação de mestrado [de Oliveira Neto 2000], seguida mais adiante por publicações no Simpósio Brasileiro de Informática na Educação [de Barros et al. 2004, Medeiros et al. 2007]. No Workshop de Educação em Computação, os padrões pedagógicos também são abordados em alguns trabalhos [de Barros e Delgado 2006, dos Santos Júnior et al. 2009]. De forma geral, nota-se que esses trabalhos em âmbito nacional não propõem ou avaliam padrões, mas sim ferramentas ou estratégias de apoio ao aprendizado com base em padrões pedagógicos.

Um aspecto importante sobre os padrões pedagógicos é seu formato de descrição, que é sempre sucinto e organizado para facilitar o reuso. Esse formato não é rígido, mas comumente inclui as seções: nome do padrão, objetivo/motivação, aplicabilidade, estrutura, consequências, implementação, recursos necessários, exemplos e padrões relacionados [Sharp et al. 1996]. Alguns autores organizam a descrição em menos seções, porém mantendo essas informações básicas [Schmolitzky 2007]. Para facilitar a comunicação, a redação do padrão geralmente usa o pronome "você" para dirigir-se diretamente ao professor.

3. O Padrão "Hall of Fame/Shame"

O padrão proposto é descrito a seguir, usando um formato semelhante ao utilizado por [Bergin 2000].

- **Nome:** HALL OF FAME/SHAME
- **Problema:** Alunos que vencem barreiras iniciais no aprendizado de programação ficam satisfeitos quando conseguem resolver novos problemas e seus programas funcionam. No entanto, esses alunos não costumam ser críticos quanto a seus códigos, ou julgam-se incapazes de produzir algo melhor no tempo disponível. Muitas vezes, seus programas são resultado de tentativas e erros, revelando más práticas de programação que são repetidas sucessivamente. Os alunos recebem orientações sobre boas práticas, porém não as relacionam com os problemas que devem resolver. O *feedback* fornecido para esses alunos ocorre muitas vezes sob

[1] http://www.pedagogicalpatterns.org
[2] http://educationalpatterns.org

forma de uma nota, um comentário ou um gabarito, que não estimulam o aluno a vislumbrar diferentes alternativas de soluções, algumas melhores que outras.

- **Contexto:** Você está ensinando um novo tópico de programação a alunos que já adquiriram alguma experiência com uma linguagem (tipicamente, alunos de segundo ano de cursos de graduação em Computação). Por exemplo, podem ser novas metodologias e técnicas de programação, um novo paradigma ou uma nova linguagem. Você requer que os alunos produzam código, de diferentes tamanhos, para resolver problemas propostos. Seu grupo de alunos tem de 20 a 40 indivíduos (não é uma turma muito pequena, nem muito grande). Você deseja estimulá-los a conhecer e adotar boas práticas, aproveitando também suas experiências anteriores em programação.
- **Solução:** Depois que o grupo de alunos entregar suas soluções para um dado problema, analise os códigos buscando identificar boas e más práticas de programação. Preferencialmente, permita que os alunos entreguem soluções parciais antes da entrega final. A cada entrega, escolha trechos de códigos para formar uma exposição, separada em duas "galerias": *Hall of Fame* (boas práticas) e *Hall of Shame* (más práticas). Em aula, para cada aspecto da solução, apresente e comente os trechos em ambas as galerias, sem revelar os autores, discutindo com os alunos os motivos para classificar cada trecho em uma ou outra galeria. Apresente também trechos de código sem classificação e discuta com os alunos o enquadramento como *Fame* ou *Shame*. Use recursos visuais contrastantes para identificar ambas as galerias. Ressalte que, num mesmo programa, é possível encontrar exemplos de boas e más práticas, reforçando que a galeria não é de bons ou maus programadores, mas sim de trechos de código com bons e maus exemplos. Faça com que a galeria fique disponível para consultas futuras. Repita este padrão sempre que possível.
- **Discussão:** Este padrão provê *feedback* aos alunos com base em problemas que eles se dedicaram a resolver. Ao contrário de recomendações gerais e abstratas sobre boas práticas, o padrão garante exemplos reais, provenientes do próprio grupo de alunos. Com a separação clara e repetitiva entre *Fame/Shame*, a intenção é estimular processos cognitivos que, nas próximas atividades de programação, resultem na adoção de uma boa prática ou, no mínimo, alimentem o senso crítico dos alunos frente a seus programas. Este padrão não visa substituir uma avaliação individual dos programas (que, por outro lado, nem sempre é possível), mas constitui uma alternativa para prover *feedback* de forma rápida e proveitosa para o grupo. O padrão tem também limitações, a saber: (i) é necessário um tempo considerável para preparar e expor/comentar as galerias, imediatamente após a entrega das soluções pelos alunos; (ii) em grupos de alunos com experiências muito homogêneas, a diversidade de exemplos produzidos pode ser pequena.
- **Recursos necessários:** Para exibição das galerias, é necessário projetor multimídia para uso em aula, além de infraestrutura para disponibilizar o material para consulta futura. Os problemas propostos aos alunos devem ser especificados de tal forma a dar margem suficiente para que se produzam soluções diversificadas. O apoio de um monitor ou assistente é recomendável na preparação das galerias, para que a exibição ocorra o mais rápido possível, idealmente em um encontro subsequente à entrega das soluções.
- **Exemplos:** Um exemplo de aplicação do HoFS é no ensino de programação procedimental em linguagem C, depois que os alunos já souberem estruturar o

código em funções e procedimentos. Neste ponto, é comum que os alunos tenham que programar um jogo simples, ou um editor de textos. A galeria *Shame*, por exemplo, possivelmente irá revelar trechos de código com os seguintes problemas: divisão do código em poucos subprogramas, procedimentos que realizam mais ou menos instruções do que o nome sugere, procedimentos que acessam variáveis globais quando deveriam se ater a seus argumentos, código mal endentado, etc. Na galeria *Fame*, ficarão os trechos de código com características positivas, opostas a essas.

Outro exemplo de aplicação é no ensino de programação funcional, em linguagens Haskell ou Lisp. Habitualmente, propõem-se que os alunos resolvam vários tipos de problemas com listas, envolvendo percursos recursivos ou via funções de ordem superior (*higher order functions*). Quando os alunos já conhecem ambas as alternativas, a galeria *Shame* poderá apontar, por exemplo, trechos de código que usaram recursão explícita para implementar o que seria mais facilmente escrito e reusável usando funções de ordem superior, como `map` ou `filter`.

- **Padrões relacionados:** A aplicação do padrão HoFS supõe que os alunos devam produzir código para implementar algum software, mesmo que parcialmente. Assim, padrões relacionados ao ensino de desenvolvimento de software podem ser aliados no processo de ensino-aprendizagem [Schmolitzky 2007]. Além disso, o padrão HoFS é centrado na ideia de que os alunos se beneficiam de *feedback* frequente para progredirem e desenvolverem suas habilidades em programação. Desta forma, outros padrões pedagógicos que também focam em *feedback* contínuo [Larson et al. 2008, Köppe et al. 2015] podem complementar a aplicação do HoFS.

4. Aplicação e Avaliação do HoFS

O padrão aqui apresentado foi aplicado em 4 semestres de uma disciplina que trata de Programação Orientada a Objetos (POO), oferecida na Universidade Federal de Santa Maria, para alunos cursando a partir do terceiro semestre dos cursos de graduação em Paradigmas de Programação. Essa disciplina proporciona um primeiro contato com POO aos alunos e, para isso, adota a linguagem Java. Observou-se, nas primeiras instâncias de aplicação, que o padrão pareceu ter aceitação positiva por parte dos alunos, evidenciada principalmente por: (i) uma maior atenção e participação dos alunos em sala de aula durante a exposição dos códigos, em contraste com exposições de conceitos ou trechos de código do professor e (ii) interesse de alguns alunos em saber se seus códigos entrariam ou não para as galerias.

Diante dessas observações, na última oferta da disciplina buscou-se realizar uma avaliação do padrão HoFS com critérios e procedimentos previamente estabelecidos, visando coletar mais dados sobre seu impacto no aprendizado de POO. O restante desta seção se refere à avaliação do HoFS aplicado à última turma.

Na literatura sobre padrões pedagógicos, não se encontram critérios e procedimentos sistematicamente utilizados para avaliar padrões propostos. Assim, decidiu-se avaliar 2 aspectos complementares que, em nossa experiência, poderiam fornecer indicadores sobre a eficácia (ou não) do padrão. Um desses aspectos foi a **aceitação** da abordagem por parte dos alunos, avaliada por meio de um questionário respondido ao final da disciplina. Outro aspecto foi a **produção** de código pelos alunos, contendo bons e/ou maus exemplos de práticas de programação, avaliada pela análise continuada do código produzido para

resolver problemas propostos. Entende-se que esses aspectos são complementares pois, por exemplo, os alunos poderiam avaliar positivamente a experiência, mas suas produções de código não revelarem melhoria de suas práticas.

4.1. Avaliação da aceitação: método e resultados

Para avaliar a aceitação, elaborou-se um questionário dividido em 2 partes, cada parte apresentada sob forma de itens da escala de Likert de 5 pontos, variando de -2 (discordo totalmente) a +2 (concordo totalmente). A primeira parte continha 5 questões relacionadas à percepção geral do padrão pelos alunos, relacionando-o com outras experiências de aprendizado. A segunda parte apresentava 4 questões sobre o formato utilizado para exibição e discussão das galerias, que compreendeu aulas expositivas e participativas usando *slides* com trechos de códigos, seguida de publicação do material *online* para consultas futuras (conforme apresentado na seção 3). Nessa segunda parte, buscou-se levantar a necessidade de possíveis ajustes na aplicação do padrão.

O questionário foi respondido anonimamente por 15 alunos, num total de 22 que cursaram a disciplina até o final. Esses alunos tiveram aulas teóricas e práticas, realizaram exercícios e desenvolveram 2 trabalhos/projetos individuais de POO. O padrão HoFS foi aplicado em 3 oportunidades: (1) após a entrega do primeiro trabalho, (2) após uma entrega parcial do segundo trabalho (50% dos requisitos implementados) e (3) após a entrega final do segundo trabalho. A avaliação dos alunos na disciplina considerou seu desempenho nos trabalhos e em uma avaliação escrita. O questionário foi apresentado após a divulgação de notas, ao fim do semestre.

A figura 1 apresenta os resultados obtidos na primeira parte do questionário. Pode-se notar que uma das questões continha uma proposição com viés negativo (“Achei ruim ver meus erros expostos, mesmo anonimamente”). De forma geral, as respostas dos alunos reforçaram a hipótese inicial sobre a boa aceitação da abordagem. Além disso, os alunos perceberam um caráter original na abordagem, ao não concordarem com a afirmação “Já tive aulas assim em outras disciplinas”.

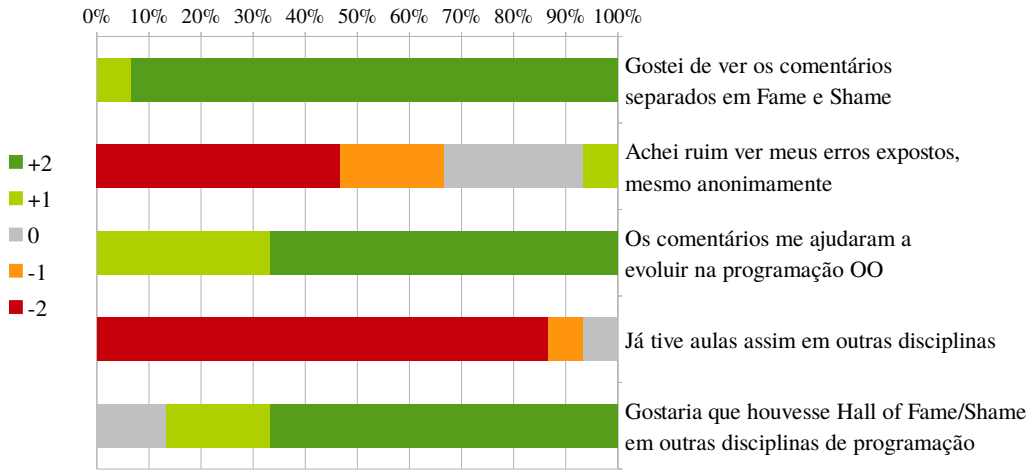

Figura 1. Avaliação da percepção geral do padrão pelos alunos

Na figura 2, apresenta-se as respostas para as questões sobre o formato de aplicação do padrão. Neste quesito, de forma geral, nota-se também que houve boa aceitação do

formato, porém alguns itens merecem mais atenção: (1) os comentários *Fame/Shame* **antes** da entrega final de um trabalho foram entendidos como mais úteis do que aqueles apresentados **após** a entrega; (2) muitos alunos responderam "Concordo parcialmente" ao item "O material disponibilizado para consultas posteriores foi suficiente". O item (1) sugere que, quando possível, o padrão seja aplicado enquanto os alunos ainda estão programando suas soluções. O item (2), por sua vez, revela um possível descompasso entre o aproveitamento das galerias durante e após as aulas. Os motivos disso podem ser investigados em novas experiências mas, como alternativa imediata, estima-se que a disponibilização das galerias em formato de vídeo poderia reduzir esse descompasso.

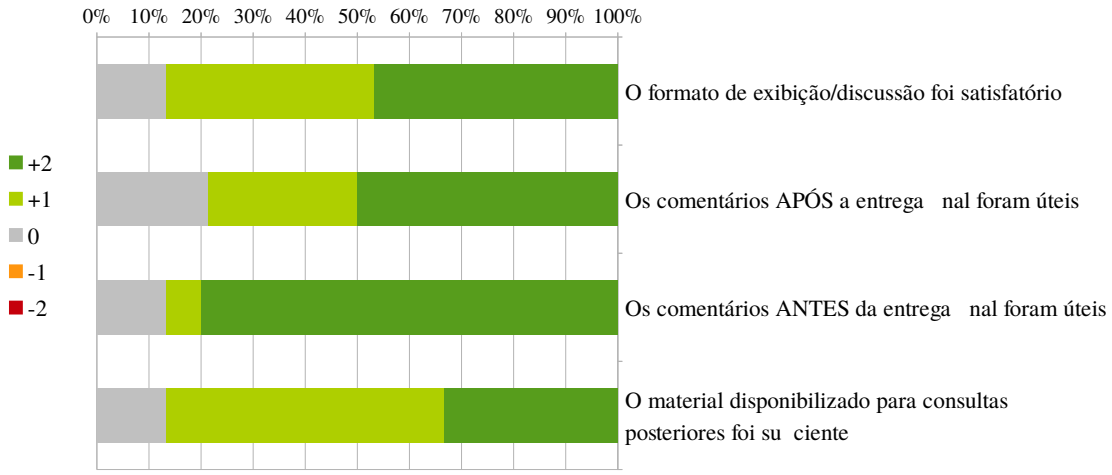

Figura 2. Avaliação do formato adotado para as aulas que seguiram o padrão HoFS

4.2. Avaliação da produção: método e resultados

Para avaliar o impacto do HoFS na produção dos alunos, considerou-se 2 trabalhos/projetos de POO desenvolvidos pelos alunos. Os programas referentes a esses projetos foram criados pelos alunos no prazo de 1 e 3 semanas, respectivamente, e entregues ao professor em 3 oportunidades, conforme explicado na seção anterior.

O primeiro projeto tinha como tema a simulação de entradas e saídas de veículos num estacionamento, compreendendo o cálculo do valor a ser cobrado para cada veículo. Esse projeto não explicitava requisitos quanto à interface com o usuário, enfatizando principalmente o projeto e implementação orientados a objetos para o domínio em questão. O padrão HoFS foi aplicado após a entrega final deste trabalho.

O segundo projeto, por sua vez, consistiu em desenvolver uma aplicação com interface gráfica para *desktop*, em Java, visando gerenciar um cadastro de postos de combustíveis em um município, com seus respectivos históricos de preços. O projeto deveria utilizar o padrão MVC (*Model-View-Controller*) e persistir os dados em arquivo e/ou banco de dados. Para este trabalho, aplicou-se o padrão HoFS antes e após a entrega final.

Para análise da produção dos alunos, considerou-se todos os códigos entregues pelos alunos nos 3 momentos descritos anteriormente, num total de 17 e 20 projetos entregues, respectivamente, para o primeiro e segundo trabalhos. Para cada exposição

Fame e *Shame* em aula, escolheu-se em média 15 trechos de código para discussão, entre bons e maus exemplos, versando sobre diversos aspectos da POO.

Pelos temas dos projetos, há boas práticas de POO que não eram necessárias nos 2 trabalhos (por exemplo, classes de acesso aos dados persistidos em arquivo ou banco de dados). Por outro lado, algumas práticas relacionadas aos fundamentos da POO deveriam estar presentes em todos os trabalhos. Nessa situação, a hipótese era de que o padrão HoFS poderia estimular os alunos a corrigir suas más práticas, entre um trabalho e outro. Assim, dentre todas as boas e más práticas identificadas nos trabalhos, escolheu-se 3 más práticas cujas ocorrências foram contabilizadas nos códigos **antes** do primeiro HoFS (isto é, no primeiro trabalho entregue) e **depois** do último HoFS (isto é, no último trabalho entregue). Essas práticas foram: (1) duplicação de código ao invés de polimorfismo, (2) violação do princípio da responsabilidade única na implementação de uma classe e (3) visibilidade pública para atributos de instância.

Na tabela 1, apresenta-se as ocorrências identificadas, sendo que um aluno pode ser responsável por mais de uma ocorrência. Pode-se notar que as ocorrências destas más práticas diminuíram, como esperado. Sabe-se que os processos de ensino-aprendizagem são por natureza complexos e, por isso, não se pode afirmar com certeza que a diminuição de más práticas seja consequência unicamente da aplicação do HoFS. Mesmo assim, interpreta-se estes resultados como indícios de que o padrão pode ser eficaz. Mais importante do que isso é o evidente progresso demonstrado pelos alunos e estimulado pelo HoFS.

Má prática	Ocorrências antes	Ocorrências depois
Duplicação de código	5	1
Violação do princípio da responsabilidade única	10	2
Visibilidade pública para atributos de instância	3	0

Tabela 1. Ocorrências de más práticas nos códigos antes e depois da aplicação do HoFS

5. Considerações Finais

Neste trabalho, buscou-se caracterizar um padrão pedagógico focado no ensino de programação. Esse padrão relaciona-se com outros voltados ao *feedback* contínuo, porém difere desses à medida em que se fundamenta essencialmente na análise e exibição de trechos de código que representam boas e más práticas de programação.

O padrão foi aplicado numa instituição de ensino superior e, na última instância de aplicação, realizou-se uma avaliação baseada na aceitação do padrão pelos alunos e na análise de códigos por eles produzidos. Os resultados apontam para a eficácia da abordagem e destacam seu caráter original, embora seja necessário repetir o experimento para aferir tais afirmações.

Tem-se consciência de que, isoladamente, um padrão pedagógico não pode garantir o sucesso do processo de ensino-aprendizagem. No entanto, padrões documentados, aplicados e discutidos podem constituir recursos úteis professores que procuram diversificar e aprimorar suas práticas. Assim, como perspectivas futuras relacionadas a este trabalho, pode-se vislumbrar que mais professores também compartilhem padrões originais e/ou relatem experiências de aplicações de padrões pedagógicos em Computação.

Visando facilitar a reprodução do padrão em outros tempos e espaços, as galerias produzidas e outros materiais utilizados encontram-se disponíveis em: `http://www-usr.inf.ufsm.br/~andrea/elc117-2015b/`.

Referências

Aureliano, V. e Tedesco, P. (2012). Ensino-aprendizagem de programação para iniciantes: uma revisão sistemática da literatura focada no SBIE e WIE. In *Anais do Simpósio Brasileiro de Informática na Educação (SBIE)*.

Bergin, J. (2000). Fourteen pedagogical patterns. In *Proceedings of the Fifth European Conference on Pattern Languages of Programs*, Irsee, Germany.

Bergin, J., Eckstein, J., Volter, M., Sipos, M., Wallingford, E., Marquardt, K., Chandler, J., Sharp, H., e Manns, M. L. (2012). *Pedagogical Patterns: Advice For Educators*. Joseph Bergin Software Tools.

Caspersen, M. E. e Kolling, M. (2009). Stream: A first programming process. *Trans. Comput. Educ.*, 9(1):4:1–4:29.

de Barros, L. N. e Delgado, K. V. (2006). Aprendizado de Programação. In *Anais do Workshop sobre Educação em Computação (WEI)*, pages 31 – 40.

de Barros, L. N., Delgado, K. V., e Machion, A. C. G. (2004). An ITS for programming to explore practical reasoning. In *Anais do Simpósio Brasileiro de Informática na Educação (SBIE)*.

de Oliveira Neto, J. A. (2000). Suporte de ferramenta de software para o padrão pedagógico aula em mapa de conceitos. Master's thesis, Universidade Federal da Paraíba, Campina Grande.

Denning, P. J. (1989). A debate on teaching computing science. *Commun. ACM*, 32(12):1397–1414.

dos Santos Júnior, G. P., Fechine, J. M., e de Barros Costa, E. (2009). Analogus: Um ambiente para auxílio ao ensino de programação orientado pelo raciocínio por analogia. In *Anais do Workshop sobre Educação em Computação (WEI)*, pages 499 – 508.

Gamma, E., Helm, R., Johnson, R., e Vlissides, J. (1993). Design patterns: Abstraction and reuse in object-oriented designs. In Nierstrasz, O., editor, *Proceedings of ECOOP'93*, Berlin. Springer-Verlag.

Jenkins, T. (2002). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, pages 53–58.

Köppe, C. (2015). Towards a pattern language for lecture design: An inventory and categorization of existing lecture-relevant patterns. In *Proceedings of the 18th European Conference on Pattern Languages of Program*, EuroPLoP '13, pages 3:1–3:17, New York, NY, USA. ACM.

Köppe, C., Portier, M., Bakker, R., e Hoppenbrouwers, S. (2015). Lecture design patterns: More interactivity improvement patterns. In *Preprints of the 22nd Pattern Languages of Programs conference, PLoP*, volume 15.

Lahtinen, E., Ala-Mutka, K., e Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '05, pages 14–18, New York, NY, USA. ACM.

Larson, K. A., Trees, F. P., e Weaver, D. S. (2008). Continuous feedback pedagogical patterns. In *Proceedings of the 15th Conference on Pattern Languages of Programs*, PLoP '08, pages 12:1–12:14, New York, NY, USA. ACM.

Medeiros, F. M., Hernández-Domínguez, A., de Medeiros, F. N., e da Silva, A. G. (2007). Um sistema de ensino na web baseado no padrão pedagógico exposição teórica-exemplos-atividade-apresentação-avaliação. In *Anais do Simpósio Brasileiro de Informática na Educação (SBIE)*.

Mor, Y., Mellar, H., Warburton, S., e Winters, N. (2014). *Practical Design Patterns for Teaching and Learning with Technology*. Sense Publishers.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., e Paterson, J. (2007). A survey of literature on the teaching of introductory programming. In *Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '07, pages 204–223, New York, NY, USA. ACM.

Robins, A., Rountree, J., e Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13:137–172.

Schmolitzky, A. (2007). Patterns for teaching software in classroom. In *Proceedings of the 12th European Conference on Pattern Languages of Programs (EuroPLoP 2007)*, EuroPLoP '07, pages B5:1–B5:10, Irsee, Germany. Hillside Europe.

Sharp, H., Manns, M. L., McLaughlin, P., Prieto, M., e Dodani, M. (1996). Pedagogical patterns – successes in teaching object technology: A workshop from OOPSLA '96. *SIGPLAN Not.*, 31(12):18–21.

Soloway, E. e Spohrer, J. C. (1988). *Studying the Novice Programmer*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

Tan, P. H., Ting, C. Y., e Ling, S. W. (2009). Learning difficulties in programming courses: Undergraduates' perspective and perception. In *Computer Technology and Development, 2009. ICCTD '09. International Conference on*, volume 1, pages 42–46.

Wilson, B. C. e Shrock, S. (2001). Contributing to success in an introductory computer science course: A study of twelve factors. In *Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '01, pages 184–188, New York, NY, USA. ACM.

Winslow, L. E. (1996). Programming pedagogy – a psychological overview. *SIGCSE Bulletin*, 28(3):17–22.