Ambiente para aprendizagem de programação fundamentado em arquiteturas pedagógicas

Guilherme R. Marques, Orivaldo de L. Tavares, Renan F. Almeida, Roberto G. Morati Junior

Programa de Pós-Graduação em Informática – Universidade Federal do Espírito Santo (UFES) – Caixa Postal 01.9011 – 29.075-910 – Vitória – ES – Brasil

guilherme_rm@yahoo.com.br; tavares@inf.ufes.br; ralmeida@ifes.edu.br; robertomorati@gmail.com.

Abstract. *In order to contribute for the opportunities to promote learning programming, based on different solutions and based on the interaction and collaboration among students, this paper presents the design of an environment based on the Pedagogical Architecture, called Environment for Learning Programming. This article focuses on contributions that directly impact the programming learning, learning processes and education.*

Resumo. *De modo a contribuir com oportunidades de promover a aprendizagem de programação, baseadas em diferentes soluções e a partir da interação e colaboração entre os alunos, este artigo apresenta a concepção de um ambiente, baseado em Arquiteturas Pedagógicas, chamado Ambiente de Aprendizagem de Programação. Esse artigo busca contribuições que impactam diretamente na aprendizagem de programação, processos de aprendizagem e educação.*

1. Introdução

Mesmo que diante de muitos estudos, a aprendizagem de programação continua sendo um grande desafio. A dificuldade dos alunos na aprendizagem de programação está relacionada com várias habilidades requeridas ao aluno e que não fazem parte de suas experiências prévias, conforme discutido em [Chagas e Oliveira 2011]

O esforço empreendido na construção de algoritmos e programas tende a se transformar em obstáculo que resulta em situações problemáticas, tais como: alto índice de retenção; acúmulo de dificuldades que influenciam nos níveis de evasão nos cursos; dificuldades nas disciplinas diretamente dependentes das habilidades de programar; dificuldades em dominar o raciocínio lógico e em resolver problemas [Chagas e Oliveira 2011].

Diante do exposto, são necessárias mudanças nas estratégias e aumento da eficiência da mediação do professor com o uso de tecnologias para apoiar o aluno na construção dessas habilidades. Assim, o professor como mediador do processo de ensino-aprendizagem precisa possuir a percepção dos pontos de dificuldades dos alunos, de modo a poder intervir formativamente e fornecer *feedback* ao aluno [Chagas e Oliveira 2011].

Outra possibilidade de auxiliar os estudantes é o desenvolvimento de competências de autoavaliação por parte deles, como forma de apoiá-los na compreensão de problemas e elaboração das estratégias, de modo a reduzir a lacuna entre o desempenho alcançado e o esperado [Sadler 1998].

DOI: 10.5753/cbie.sbie.2016.1275

A partir de resultados obtidos por [Menezes e Castro Junior 2003] em programas de formação a distância, percebe-se que o uso de estratégias pedagógicas e ambientes virtuais adequados podem possibilitar novas experiências não suportadas nas salas de aula tradicionais e da educação estritamente presencial.

Sendo assim, com o intuito de apresentar contribuições no processo de aprendizagem de programação, este artigo apresenta o Ambiente de Aprendizagem de Programação (AAP), suportado por um conjunto de Arquiteturas Pedagógicas (APs). O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta trabalhos correlatos sobre pesquisas correlatas. Na Seção 3, são apresentadas as Arquiteturas Pedagógicas fundamentados neste trabalho como suporte teórico. A Seção 4 apresenta o ambiente AAP. A Seção 5 apresenta resultados alcançados. Por fim, a Seção 6 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos correlatos

A pesquisa apresentada por [Chagas 2014] busca construir um ambiente de apoio à aprendizagem de programação suportadas por APs. Uma das APs apresentada é a CSPL (*Computer Supported Programming Learning* – Computadores como suporte à aprendizagem de programação), construída com base no uso do MPC (Método para Construção de Programas). O MCP, descrito em [Tavares *et al.* 2013], estrutura a construção de um programa em seis etapas: 1) compreensão do problema; 2) planejamento de testes; 3) especificação da solução em linguagem natural; 4) codificação do programa; 5) testes da solução construída com base no planejamento da etapa 2 e 6) avaliação do processo como um todo.

A partir da pesquisa citada, percebeu-se uma carência da AP citada em permitir ao aluno explorar e interagir com possíveis soluções elaboradas pelos seus colegas, além de não armazenar o percurso de aprendizagem dos estudantes para acompanhamento do processo de aprendizagem dos alunos em aulas de programação. Sobre o suporte computacional da pesquisa citada, percebeu-se também uma carência de mecanismos que facilitasse a recuperação de informações para avaliação dos processos de aprendizagem individual e colaborativo dos alunos, bem como flexibilizar a aplicação das mesmas em uma turma com grande volume de alunos.

Desta forma, vislumbrou-se a necessidade de aprimorar as APs citadas em [Chagas 2014], de modo a apoiar a concepção do ambiente AAP e possibilitar uma melhor avaliação da eficácia das APs.

3. Arquiteturas Pedagógicas para aprendizagem de programação

Segundo [Tavares *et al.* 2013], os programas a serem construídos pelos aprendizes de programação admitem muitas possíveis soluções, porém cada uma delas possuem necessidades comuns. Quando um aprendiz constrói uma solução e se depara com as soluções dos colegas, ele pode exercitar uma reflexão sobre as alternativas possíveis e sobre as necessidades comuns a todas as soluções. Apoiado por [Piaget 1985], que define que o “possível” constitui-se o produto de uma construção do sujeito em interação com as propriedades do objeto, inserindo-as em interpretações várias devido às atividades do sujeito, o que determinará, simultaneamente, a abertura de possíveis cada vez mais numerosos [Novaes 1987]. Ainda, em pesquisas realizados por [Carvalho *et al.* 2005] sobre APs, por [Tavares *et al.* 2012] e [Sirotheau *et al.* 2011] serviram de base para o estudo e aprendizado de programação para a fundamentação das APs.

Diante do exposto, esta seção apresenta APs para suportar o aprendizado de programação. A estrutura de uma AP adotada, definida por [Reinoso e Tavares 2015], é composta por objetivo pedagógico (o que aprender), atividades pedagógicas (o que fazer), método pedagógico (como fazer) e recursos tecnológicos que viabilizam a execução das atividades planejadas.

3.1. AP Programação em Pares

Segue a descrição da AP Programação em Pares:

- Objetivo pedagógico: explorar a aprendizagem de programação em pares, desenvolver o espírito crítico nos alunos ao explorarem várias possíveis soluções e explorar a aprendizagem em uma construção colaborativa de solução.
- Atividades: construir uma solução para um exercício proposto pelo professor; em dupla, revisar e testar a solução de seu par; em dupla, os alunos constroem uma terceira solução diferente das duas soluções anteriores da dupla.
- Método: 1) o professor propõe um exercício, ou lista de exercícios, para os alunos; 2) cada aluno constrói uma solução para o exercício proposto e 3) a envia para o professor; 4) pares de alunos serão formados; 5) cada membro de um par revisa a solução do seu colega. Após a revisão, 6) cada par deve construir colaborativamente uma nova solução para o exercício, diferente das soluções construídas anteriormente pelos membros do par, e enviam a solução para o professor. A Figura 1 apresenta o método descrito.
- Recursos tecnológicos: armazenador de exercícios ou listas de exercícios para acesso dos alunos; um editor de soluções seguindo as etapas do MCP; um visualizador que permita ao professor ver cada solução dos alunos; uma ferramenta para formar duplas; e um editor de soluções seguindo a MCP que permita a dois alunos construírem colaborativamente uma solução; uma ferramenta para restringir a entrega de soluções similares às soluções anteriormente entregues pelo aluno.

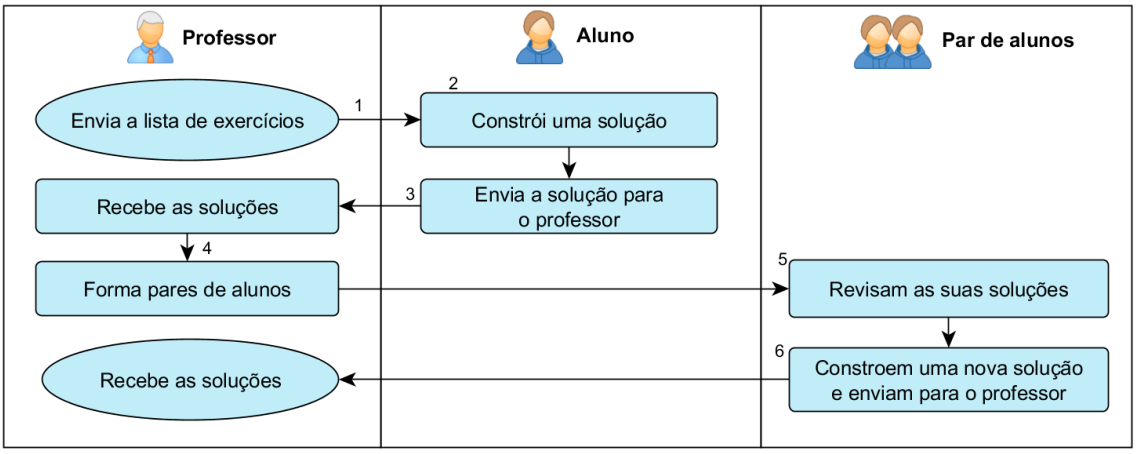

Figura 1. Atividades pedagógicas da AP Programação em Pares

3.2. AP Visita à Possíveis Soluções

Segue a descrição da AP Visita à Possíveis Soluções:

- Objetivo pedagógico: desenvolver o espírito crítico do aluno frente a possíveis

soluções;

- Atividades: cada aluno constrói uma solução para um problema; cada aluno revisa as soluções de dois de seus colegas e insere comentários seguindo critérios descritos pelo professor;
- Método: 1) o professor propõe um exercício ou uma lista de exercícios. 2) Cada o aluno constrói uma solução, e 3) envia a sua solução para o professor; 4) o professor compartilha cada solução recebida com dois alunos; 5) o professor disponibiliza para os alunos critérios a serem usados para revisarem as soluções dos colegas compartilhadas; 6) cada aluno deve revisar as soluções dos colegas e 7) inserir comentários seguindo os critérios descritos pelo professor. A Figura 2 apresenta o método descrito.
- Recursos tecnológicos: armazenador de exercícios ou listas de exercícios para acesso dos alunos; um editor de soluções seguindo as etapas do MCP; um visualizador que permita ao professor ver cada solução dos alunos; uma ferramenta para compartilhar cada solução recebida com dois alunos; um mural das soluções dos alunos, onde cada aluno é capaz de visitar as soluções dos colegas; um quadro de destaque onde o professor descreve os critérios a serem seguidos para análise da solução do colega; um testador de solução para que o aluno possa realizar testes das soluções e que possibilite editar o plano de teste com o acréscimo de novos testes ou com a mescla com outro plano de teste; armazenador de comentários realizados pelos alunos e professores para cada solução;

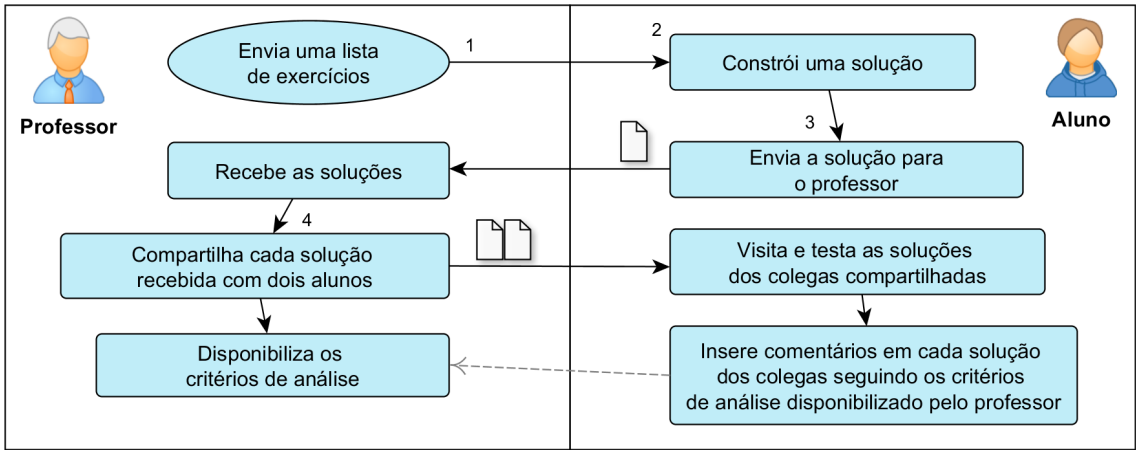

Figura 2. Atividades pedagógicas da AP Visita à possíveis soluções

3.3. AP Testa Soluções Socializadas

Nesta AP, o professor realiza uma socialização das soluções disponibilizando todas as soluções recebidas para que os alunos possam visitar, revisar e realizar outros testes. Segue a descrição da AP Testar soluções socializadas:

- Objetivo pedagógico: incentivar o aluno a desenvolver novas necessidades frente a visitação de possíveis soluções.
- Atividades: construir uma solução para um problema proposto pelo professor; o professor socializa as soluções para os alunos; após a visitação, os alunos devem construir uma solução diferente da solução envida anteriormente.

- Método: 1) o professor propõe um exercício ou lista de exercícios, 2) o aluno constrói uma solução para o exercício e 3) envia a sua solução para o professor; 4) o professor recebe as soluções dos alunos e então 5) socializa as soluções para que todos os alunos possam visitar as soluções dos colegas. Em seguida, os alunos 6) visitam as soluções dos colegas, revisam e realizam novos testes. Após a visitação das soluções socializadas, cada aluno deve 7) construir uma nova solução diferente da primeira solução enviada e 8) enviar sua nova solução para o professor. A Figura 3 apresenta o método desta AP.
- Recursos tecnológicos: armazenador de exercícios ou listas de exercícios para acesso dos alunos; um editor de soluções seguindo as etapas do MCP; um visualizador que permite ao professor ver cada solução dos alunos; um mural das soluções onde os alunos são capazes de visitar as soluções de seus colegas; um testador de solução para que o aluno possa realizar testes das soluções e que possibilite editar o plano de teste com o acréscimo de novos testes ou com a mescla com outro plano de teste; uma ferramenta para restringir a entrega de soluções similares às soluções anteriormente entregues pelo aluno.

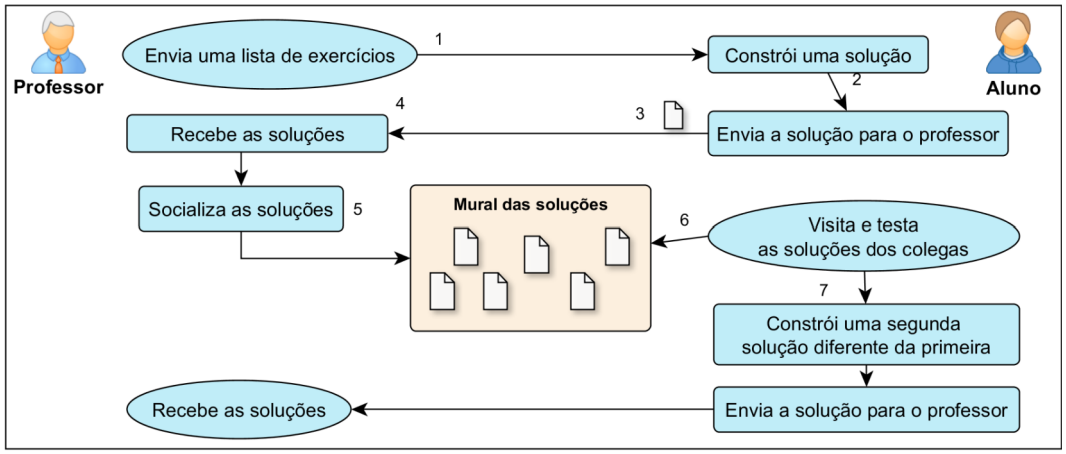

Figura 3. Atividades pedagógicas da AP Testa Soluções Socializadas.

4. AAP: Ambiente para aprendizagem de programação

A Figura 4 - Arquitetura do APP, apresenta um modelo cliente-servidor do ambiente. O AAP foi implementado com o uso das tecnologias, no lado do servidor, Apache HTTP Server, linguagem interpretada PHP, persistência de dados em banco MySQL e, no lado do cliente, as tecnologias: HTML, CSS, JavaScript, Ink Interface Kit e jQuery.

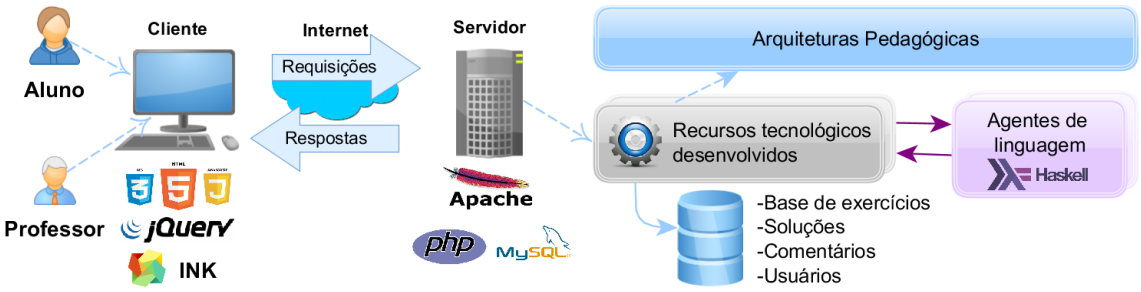

Figura 4. Arquitetura do APP

O AAP tem suporte para aprendizagem de diversas linguagens de programação disponibilizadas pela interação de agentes de software com os interpretadores ou

compiladores de cada linguagem, por exemplo, neste ambiente, utilizamos o agente de linguagem para integração com o Hugs – um interpretador Haskell.

A seguir, são apresentados os módulos que permitem a realização das APs discutidas anteriormente. Essas descrições estão resumidas devido à restrição de espaço, bem como omissão de outros módulos do ambiente.

4.1. Módulo de Administração do Professor

Este módulo admite ao professor visualizar a lista de exercícios proposta aos alunos. Além disso, o professor pode criar exercícios para compor uma base de exercícios, e elaborar novas listas de exercícios, conforme apresentado na Figura 5.

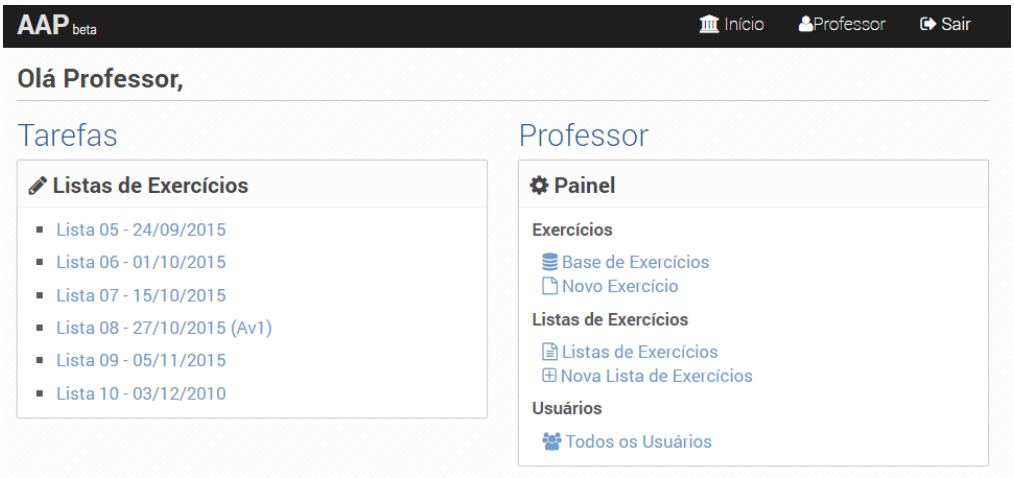

Figura 5. Módulo de Administração.

4.2. Modulo Aluno.

Admite ao aluno o acesso as listas de exercícios. O aluno pode visualizar os enunciados de todos os exercícios, acessar soluções enviadas anteriormente e acessar o Construtor de Soluções para construção de novas soluções. A Figura 6 apresenta duas telas, à esquerda os exercícios de uma lista de exercícios e à direita o Construtor de Soluções:

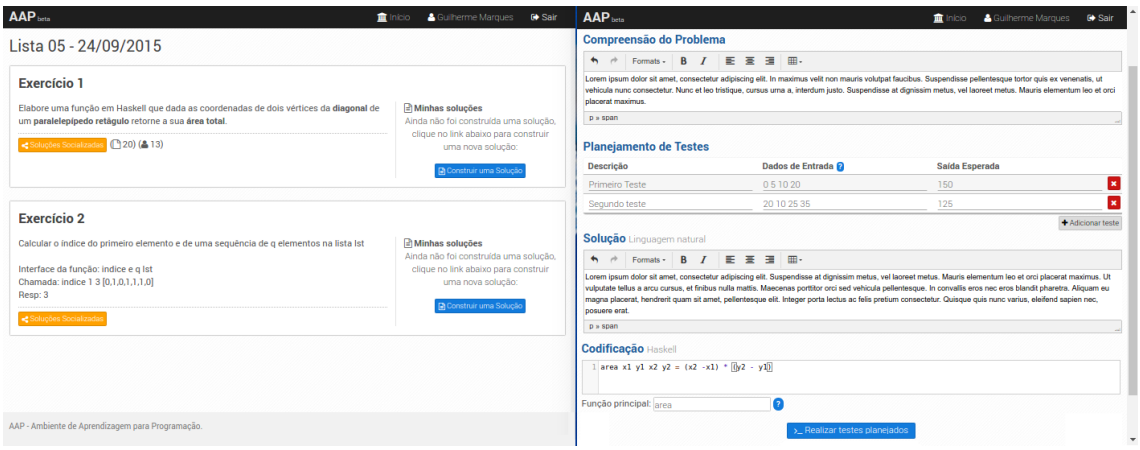

Figura 6. Módulo Aluno.

Ainda, no Módulo do Aluno, é possível a realização de testes automatizados,

conforme apresentado na Figura 7, onde os resultados dos testes são apresentados na cor verde. Caso contrário, se houver erros de execução, ou a saída esperada seja diferente da saída da execução, a linha do teste aparece destacada em vermelho para que o aluno possa verificar o erro, também são fornecidas as mensagens de saída de execução.

Figura 7. Realização de testes automáticos.

Além disso, por meio da Lista de Exercícios, o aluno pode acessar o módulo de Socialização das Soluções enviadas para um exercício, onde cada solução pode apresentar um conjunto de comentários, de modo a permitir a interação e o aprendizado colaborativo, conforme apresentado na Figura 8. É importante frisar que o acesso ao módulo de Socializações de Soluções para uma determinada lista de exercícios precisa ser liberado pelo professor.

Figura 8. Módulo de Socialização das Soluções.

Por fim, a partir das entregas das soluções dos alunos é possível que o professor gere um relatório geral com a relação de alunos e soluções entregues pelos alunos para cada exercício das listas de exercícios, conforme apresentado na Figura 9.

Figura 9. Acompanhamento de entrega das listas de exercícios.

5. Discussões e Resultados

O ambiente apresentado foi usado por uma turma de graduandos em Engenharia de Computação de uma Instituição Federal de Ensino em um período de cinco aulas em laboratório, com duração de duas horas cada aula. Apresenta-se também algumas comparações com aulas em laboratórios anteriores realizadas sem o uso do ambiente.

Nas primeiras aulas em laboratório, sem o uso do ambiente, foram propostos para os alunos quatro listas de exercícios com um total de 14 exercícios. Para cada lista era solicitado que os alunos enviassem para o e-mail do professor um relatório sobre a construção de suas soluções e testes realizados como forma de entrega dos exercícios resolvidos. A partir de um total de 49 e-mails recebidos de 32 alunos, puderam ser verificados 178 soluções para os exercícios após analisar, e-mail por e-mail, a caixa de e-mails recebidos pelo professor.

Ao final do período, em aulas em laboratório com uso do ambiente, foram enviados através do ambiente mais cinco listas de exercícios para os alunos, contendo um total de 30 exercícios no ambiente. Durante o uso do ambiente, 22 alunos construíram um total de 207 soluções. Pôde-se verificar também um total de 1528 testes realizados durante a construção das soluções a partir de consultas ao banco de dados. Esta quantidade de testes poderia ser maior se os alunos tivessem maior familiaridade com o ambiente.

Apesar de ser esperado um maior número de soluções entregues com o uso do ambiente, devido a maior quantidade de exercícios, deve-se salientar também que alguns alunos foram se dispersando durante o curso da disciplina e alguns não mais participavam das aulas.

Porém, tivemos uma avaliação positiva com o uso do ambiente, pois os alunos alegavam ser mais intuitivo o uso do ambiente para a entrega e testes das soluções construídas, sem a necessidade de trabalho para elaborar relatórios e enviá-los para o e-mail do professor, uma vez que o ambiente armazena as soluções construídas e testes realizados. Para o professor, também houve vantagens, pois o uso do ambiente facilita a localização da produção de cada aluno, e assim, evita trabalhos para analisar cada e-mail recebido dos alunos.

Seguindo as heurísticas para avaliação de usabilidade propostas por [Nielsen 1994], foi elaborado um questionário e apresentado a doze alunos no último dia de aula

em que usaram o ambiente para avaliarem a sua usabilidade. Para cada heurística, os alunos assinalaram uma nota de 0 a 10. A Figura 9 apresenta um gráfico com a média dos resultados obtidos:

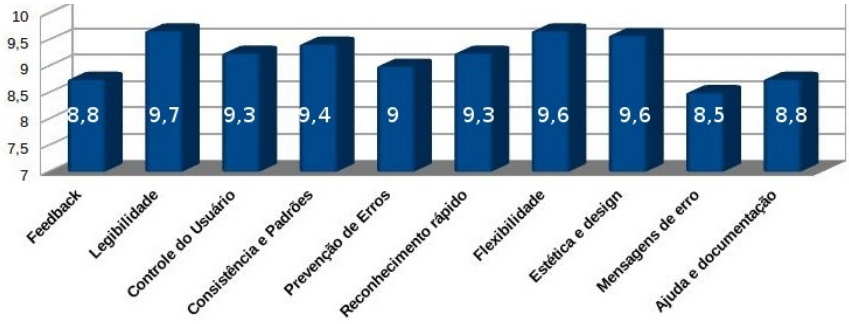

Figura 10. Resultado do questionário sobre Usabilidade.

Os resultados apresentam que as três heurísticas que tiveram menor média são: Feedback (8,8) que se refere sobre o ambiente informar ao usuário a respeito do que está acontecendo no ambiente com *feedbacks* apropriado em um tempo razoável; Ajuda e documentação (8,8) que se refere sobre o ambiente possuir um ajuda e documentação para o uso do ambiente; e Mensagens de erros (8,5) sobre o ambiente apresentar mensagens de erro que são claras o suficiente para que o usuário saiba prosseguir com sucesso ao usar ambiente.

Segundo [Nilesen 1994], deve-se preocupar com a melhoria das heurísticas que tiveram nota menor do que 6,6. Apesar das notas terem sido consideravelmente altas, elas podem se justificar devido aos alunos não terem usado, anteriormente, outras ferramentas que facilitassem, de tal forma, as atividades de programação em laboratório, e por isto, eles não possuíam experiência com outros ambientes ou ferramentas como comparativo para avaliação do APP.

6. Conclusões

Este artigo apresenta os resultados de uma abordagem que permite melhorar a aprendizagem de programação em cursos introdutórios. Uma vez que este artigo busca contribuições por meio do ambiente AAP para construção e socialização das soluções para atividades de programação. Além disso, este artigo contribui com o aprimoramento de APs, baseadas em estudos de [Piaget 1972], que possibilitaram a concepção do ambiente AAP como suporte computacional.

O ambiente AAP permite que os alunos possam visualizar listas de exercícios e construir as soluções para os mesmos seguindo a MCP. Ainda, é possível realizar testes da solução no ambiente seguindo um planejamento de teste, salvar as soluções e interagir com as soluções enviadas por outros estudantes de acordo com as atividades descritas nas APs por meio do módulo de Realização e Execução da Solução.

O ambiente AAP armazena as interações, soluções de estudantes e grupos, de modo a, permitir a recuperação de informações para conclusões e avaliações futuras dos processos envolvidos na aprendizagem dos estudantes. Conforme descrito no experimento realizado com estudantes do curso de Engenharia da Computação da Instituição de Ensino Superior (IES).

No futuro, planejamos evoluir o AAP por meio de um maior alinhamento com as APs conceituadas neste artigo. Assim, tornando possível a realização de novos

experimentos e avaliação da aplicabilidade e impactos das APs nos processos de aprendizagem de programação, buscando novos resultados a fim de continuar contribuindo com a comunidade acadêmica a respeito dos desafios com o ensino-aprendizagem de programação.

Agradecimentos

Este trabalho teve apoio e bolsa de mestrado financiado pela FAPES (Fundação de Amparo à Pesquisa e Inovação do Espírito Santo).

References

Carvalho, M. J. S., Nevado, R. A., e Menezes, C. (2005). Arquiteturas pedagógicas para educação à distância: concepções e suporte telemático. *Anais do XVI Simpósio Brasileiro de Informática na Educação, Juiz de Fora-MG. Brasil.*

Chagas, L. B. C. (2014) "Um Ambiente para Aprendizagem de Programação com o uso de Arquiteturas Pedagógicas"; Dissertação de Mestrado; PPGI/UFES; Vitória-ES, Brasil.

Chagas, L. B. C. e Oliveira, M. G. (2011). Metodologia ANEA para avaliação online de lógica de programação. In *Anais do Workshop de Informática na Escola*, volume 1, pages 1394-1397.

Menezes, C. S.; Chaves, T. H. C. e Castro Junior, A. N. (2003). Gestão do conhecimento aplicada a equipes de programação. *I Workshop de Tecnologias da Informação e Gestão do Conhecimento.*

Novaes, M. H. (1987). A evolução dos "possíveis" e dos "necessários": sua influência nos processos criativos. *Arquivos Brasileiros de Pscologia*, 39(1):3-17.

Piaget, J. (1972). Intellectual Evolution from Adolescence to Adulthood; Source: Human Development, 15:1-12.

Piaget, J. (1985). *O possível e o necessário – evolução dos possíveis na criança.* Porto Alegre, Artes Médicas.

Reinoso, L. F. e Tavares, O. L. (2005). MVLIBRAS: ambiente digital para comunidades de aprendizagem com recursos inclusivos para surdos. In *Anais do Simpósio Brasileiro de Informática na Educação*, volume 26, pages 772.

Sadler, D. R. (1998). Formative assessment: Revisiting the territory. *Assessment in education*, 5(1):77-84.

Sirotheau, S., Brito, S. R., Silva, A. S., Eliasquevici, M. K., Favero, E. L., e Tavares, O. L. (2011). Aprendizagem de iniciantes em algoritmos e programação: foco nas competências de autoavaliação. *Simpósio Brasileiro de Informática na Educação (SBIE 2011)*, 22.

Tavares, O. L., Menezes, C. S., Aragón, R., e Costa, L. B. (2013). Uma arquitetura pedagógica auxiliada por tecnologias para ensino e aprendizagem de programação. *WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI)*, 778-787.

Tavares, O. L., Menezes, C. S., e Nevado, R. A. (2012). Pedagogical architectures to support the process of teaching and learning of computer programming. In *Frontiers in Education Conference (FIE), 2012*, pages 1–6. IEEE.