UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Silvério Sirotheau Corrêa Neto

# **APRENDIZAGEM DE INICIANTES EM ALGORITMOS E PROGRAMAÇÃO:** PROPOSTA DE UM SUBSISTEMA DE *FEEDBACK* COLABORATIVO PARA AUTOAVALIAÇÃO

Belém – Pará
2012

**Silvério Sirotheau Corrêa Neto**

# **APRENDIZAGEM DE INICIANTES EM ALGORITMOS E PROGRAMAÇÃO:** PROPOSTA DE UM SUBSISTEMA DE *FEEDBACK* COLABORATIVO PARA AUTOAVALIAÇÃO

Dissertação apresentada à Banca examinadora do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Pará para obtenção do grau de Mestre em Ciência da Computação na linha de informática educativa.
Orientador: Prof. Dr. Eloi Luiz Favero

Belém – Pará
2012

Corrêa Neto, Silvério Sirotheau

Aprendizagem de iniciantes em algoritmos e programação: proposta de um subsistema de feedback colaborativo para autoavaliação/ (Silvério Sirotheau Corrêa Neto); orientador, Eloi Luiz Favero. - 2012.

100 f. il. 28 cm

Dissertação (Mestrado) – Universidade Federal do Pará. Instituto de Ciências Exatas e Naturais. Programa de Pós-Graduação em Ciência da Computação. Belém, 2012.

1. Programação Lógica. I. Favero, Eloi Luiz, orient. II. Universidade Federal do Pará, Instituto de Ciências Exatas e Naturais, Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 22. ed. 005.115

**Silvério Sirotheau Corrêa Neto**

# **APRENDIZAGEM DE INICIANTES EM ALGORITMOS E PROGRAMAÇÃO:** PROPOSTA DE UM SUBSISTEMA DE *FEEDBACK* COLABORATIVO PARA AUTOAVALIAÇÃO

Dissertação apresentada à Banca examinadora do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Pará para obtenção do grau de Mestre em Ciência da Computação na linha de informática educativa.

**Aprovado em: 25 / 05 / 2012**

**Conceito: APROVADO**

**BANCA EXAMINADORA:**

**Prof. Dr. Eloi Luiz Favero**
**(Orientador)**
**(PPGCC / UFPA)**

**Prof. Dr. Bianchi Serique Meiguins**
**(PPGEE / UFPA)**

**Profa. Dra. Marianne Kogut Eliasquevici**
**(FACOMP / UFPA)**

**Prof. Dr. Orivaldo Lira Tavares**
**(DI / CT / UFES)**

A Deus, aos meus pais Valdomiro e Fátima, à minha esposa Ana Paula e a meu filho João Claudio e a todos pelo amor, carinho, confiança e dedicação para que eu chegasse até esse momento.

Silvério Sirotheau

## AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter nos dado a vida, saúde e paz.

À minha família, em especial à minha amada e companheira, Ana Paula Sirotheau, pelos momentos difíceis e superados, por todo o amor, carinho e dedicação, pelo filho lindo e maravilhoso. A meu filho João, pela capacidade de me fazer sorrir, chorar, refletir e amar, mesmo em dias de tempestade. Pela capacidade de mostrar um lado lindo da vida, que é de ser pai. A Meus pais, Valdomiro e Maria de Fátima Sirotheau, por estar sempre ao meu lado e acreditar que eu posso realizar meus sonhos, pelas noites perdidas trabalhando, pelas dificuldades que já passaram pra que nós, filhos, nunca sentíssemos necessidades e privações já vividas por eles. Pela capacidade de contornar os obstáculos da vida e passar por todos dando exemplo de perseverança e temor de DEUS. Às minhas irmãs, Glayce e Glauce Sirotheau, pela torcida e apoio moral diante dos obstáculos. Ao meu cunhado Alessandro Cardoso pela confiança e pensamento positivo. Aos afilhados Gabrielly, Luana e "Davi" pelas alegrias e futuras alegrias. A meu Sogro José Claudio Pinheiro do Nascimento e minha Sogra Janete Socorro Santos Raiol pela confiança. E por todos que passaram pelo meu caminho durante todo esse processo.

Agradeço ao Professor Eloi Luiz Favero, pela amizade, ao longo desse tempo, pela confiança, pela oportunidade de crescer como estudante e pesquisador.

Agradeço, também, ao Professor José Miguel Martins Veloso, por sempre acreditar nas pessoas que um dia possam mudar a educação desse mundo.

Agradeço, à Professora Marianne Kogut, pela parceria científica e ao Professor Orivaldo de Lira Tavares, pelas orientações na minha apresentação no XXII SBIE em Aracaju/SE.

Aos velhos amigos de sempre estiveram comigo: Victor Batista Bezerra e Marcelo Munhoz. Aos amigos da Faculdade de Matemática e do Programa de Pós-Graduação onde tudo começou, obrigado pela força: Maria Leonarda, Macelene, Judson, Ewerton Iverson, Silza Helena, Rose, Thainan, Marcos, Rafael Raiol e aos Professores Marcos Monteiro Diniz, Cristina Lúcia Dias Vaz, Marcio Nascimento, José Antônio Vilhena, Irene Castro Pereira, Jerônimo Noronha Neto, Midori Makino, Adam Oliveira da Silva, Augusto César Costa, Francisco Paulo Marques Lopes, Geraldo Mendes de Araújo (obrigado pela força!), João Pablo Pinheiro, Tânia Valdivia, João Claudio Brandemberg, João Carlos Alves e Manoel

Silvino (obrigado pela oportunidade!), Batalha. Agradeço a todos pelo pensamento positivo em direção ao meu objetivo.

Aos amigos que construi na AEDI pela nova conquista: Iza Helena Travassos, Ivanete Guedes, Renaceli, Marcelo, Paulo, Ana Progênio, Profa. Susane Fernandes e Profa. Joaquina Barata.

Aos parceiros do Laboratório de Multimídia pelas contribuições: Profa. Maria Ataide Malcher, Fernanda Chocron, Caio Arapiraca e Guillermo Aguilera.

Aos amigos do LabEad e do Evoox, em especial: William da Silva, Diego Hortêncio, Lucas Damasceno, Walmir Portal, Wilcley da Silva, André Cruz e Antônio Ito Hidaka. Aos amigos de conselhos e de pensamentos positivos: Silvana Rossy, Leka, Pedro Silvestre, Marcelly Mota e Antonio Lobato.

Obrigado aos professores do PPGCC que contribuíram para minha formação e a todas as pessoas que ajudaram direta ou indiretamente.

*Silvério Sirotheau*

*“Quando não houver saída*
*Quando não houver mais solução*
*Ainda há de haver saída*
*Nenhuma ideia vale uma vida...*
*Enquanto houver sol*
*Enquanto houver sol*
*Ainda haverá...”*

Titãs

*“Sim, Deus é fiel, para cumprir*
*Toda promessa feita a mim*
*Deus é fiel, Deus é fiel*

*Eu não morrerei*
*Enquanto o Senhor não cumprir em mim*
*Todos os sonhos*
*Que Ele mesmo sonhou pra mim”*

Nani Azevedo

## RESUMO

O presente trabalho descreve a proposta de um subsistema de *feedback* colaborativo para um ambiente de programação integrado à plataforma *Moodle*. Tal subsistema objetiva auxiliar estudantes em processo de aprendizagem, por proporcionar recursos de autoavaliação que são empregados durante a elaboração de códigos de programas, incentivando a argumentação, a exposição de ideias e a resolução de conflitos entre os estudantes. Como vantagem do subsistema de *feedback* visualiza-se: (1) amenizar os problemas de compreensão dos códigos de programas elaborados pelos estudantes; (2) prover um retorno imediato ao estudante sobre soluções de sua resposta por meio de um procedimento de avaliação entre os estudantes, chamada de avaliação em pares; e (3) diminuir a sobrecarga de trabalho do professor. A proposta foi avaliada a partir de um procedimento com três passos: (a) a oferta de um curso de programação; e (b) avaliação heurística; (c) ensaio de interação. Os resultados provenientes da avaliação permitiram realizar melhorias no ambiente de programação com o *feedback* colaborativo, contribuindo para compreensão das abstrações dos códigos e ao mesmo tempo reduzindo o espaço de tempo do *feedback*. Para o professor, o subsistema oportuniza a adoção de novas abordagens, porque fornece um mapa com resultados, ainda que parciais, do desempenho dos estudantes. Por um lado, a prática com atividades colaborativas deve contribuir para a identificação de requisitos para novas funcionalidades, como a programação em pares e, por outro lado, deve estimular o desenvolvimento da autoavaliação, de forma que, possa proporcionar um aumento no nível das habilidades dos estudantes em programação.



Palavras-Chave: ensino de programação, autoavaliação da aprendizagem, *feedback*.

**ABSTRACT**

This paper describes a proposed subsystem feedback to a collaborative programming environment integrated with Moodle. This subsystem aims to assist students in the learning process by providing self-assessment resources that are employed during the development of codes of programs, encouraging discussion, exhibition of ideas and conflict resolution among students. How advantage subsystem feedback is visualized: (1) reduce the problems of understanding the codes of programs designed by the students, (2) provide immediate feedback to the student about their response solutions through an evaluation procedure among students , called peer evaluation, and (3) reduce the workload of teachers. The proposal was evaluated using a procedure with three steps: (a) the provision of a programming course, and (b) heuristic evaluation, (c) test interaction. The results from the evaluation allowed to make improvements in the programming environment with collaborative feedback, contributing to understanding the abstractions of the code while reducing the time of feedback. For the teacher, the subsystem favors the adoption of new approaches, it provides a map with results, even partial, of student performance. On one hand, the practice with collaborative activities should contribute to the identification of requirements for new features such as programming in pairs and, moreover, should stimulate development as perceived, so that could provide an increased level of skills students in programming.

Keywords: educational programming, self-assessment of learning, feedback.

# LISTA DE ILUSTRAÇÕES

# SUMÁRIO

# 1 INTRODUÇÃO

A primeira experiência no aprendizado de programação, para muitos acadêmicos da área de computação, constitui-se em preocupação crescente nas instituições de ensino superior. Essa preocupação fundamenta-se em indicadores que medem a evasão, o insucesso, o tipo de atuação profissional esperada do egresso e o nível de insatisfação com o curso, além de outros. Entre os motivos que justificam o fracasso no ensino e na aprendizagem de programação, ressaltam-se (BORGES, 2000 ; GIRAFFA et al, 2003):

- hábitos de estudo pouco disciplinados e centrados em memorização;
- conhecimentos prévios incipientes, principalmente nos domínios matemáticos e lógicos;
- abordagens pouco motivadoras;
- conteúdo pouco relacionado ao cotidiano dos estudantes;
- dificuldade na compreensão do enunciado dos problemas;
- forte carga de conceitos abstratos.

Por esses e outros motivos, Borges (2000) alega que tanto é alto o índice de estudantes que completam o curso sem ter um conhecimento mínimo adequado na área de computação, como também é alto o índice de abandono nos primeiros semestres do curso, devido às dificuldades em programação. Proulx (2000) afirma que quando o estudante não é bem sucedido durante os primeiros contatos com a programação poderá ter problemas no futuro, sobretudo, dificuldades demonstradas nas disciplinas diretamente dependentes das habilidades de programar, dominar o raciocínio lógico e resolver problemas. Em alguns casos, quando o estudante percebe que não está evoluindo na disciplina, sente-se desmotivado e seu desempenho torna-se cada vez menor (ALMEIDA, 2002).

As habilidades necessárias para a construção de programas de computadores exigem empenho significativo por parte dos estudantes e um grande esforço de acompanhamento do professor que, muitas vezes, para minimizar essas dificuldades, dedica vigor no ensino dos aspectos sintáticos e semânticos das linguagens de programação. Esta abordagem didática, com frequência, envolve a apresentação de exemplos com programas de estruturas semelhantes àquelas a serem aplicadas nesses primeiros exercícios de programação. Segundo Menezes et al. (2008), tal estratégia

está fundamentada no fato de que os estudantes precisam criar, em suas memórias, padrões de estruturas que possam ser usadas para resolver diferentes tipos de problemas.

Outra maneira de auxiliar os estudantes a melhorar sua capacidade de programar pode estar no desenvolvimento da autoavaliação por parte destes, como forma de apoiá-los na compreensão dos objetivos da disciplina, no seu desempenho e nas suas estratégias utilizadas para reduzir ou eliminar a lacuna entre o desempenho alcançado e o esperado (SADLER 1998). Diante deste contexto, este trabalho defende uma prática autoavaliativa em um Ambiente de Aprendizagem de Programação *Java* integrado à Plataforma Virtual de Ensino *Moodle* (MOTA et al., 2009), por meio de recursos que possibilitam o exercício do *feedback* entre estudantes, como maneira de estimular o desenvolvimento de habilidades de avaliação. A proposta está baseada na especificação de novos requisitos no ambiente de programação já existente e de uma revisão na literatura em torno dos problemas no ensino-aprendizagem de programação e de autoavaliação.

## 1.1 MOTIVAÇÃO

As dificuldades dos estudantes em assimilar conceitos preliminares de programação provocam um transtorno na caminhada acadêmica destes, sendo determinantes para resultados indesejados já referenciados. Além destes, a alta taxa de evasão de estudantes e a dificuldade de acompanhamento pelo professor nas atividades de prática no laboratório, são alguns dos fundamentos motivacionais para o desenvolvimento desse trabalho.

Um dos recursos para diminuir o problema enfrentado pelos estudantes é a proposta deste trabalho, em desenvolver um subsistema que possibilite o exercício do *feedback* colaborativo (Avaliação em pares) entre estudantes, como forma de estimular o desenvolvimento de habilidades de autoavaliação. Por meio deste, docentes podem manter ou reformular suas práticas sobre programação; estudantes podem melhorar soluções de outros estudantes dinamizando potencialidades e solucionando problemáticas condizentes ao método de aprendizagem em programação. Na literatura da área não foi encontrado uma ferramenta similar que auxilie nesse processo inicial em programação.

## 1.2 OBJETIVO

Este trabalho possui como objetivo geral conceber e desenvolver um subsistema de *feedback* colaborativo num ambiente de aprendizagem de programação incorporado ao *Moodle*, já existente no intuito de auxiliar na construção de habilidades possibilitando a autoavaliação entre os estudantes.

### 1.2.1 Objetivos específicos

- integrar o subsistema de *feedback* colaborativo com o ambiente de programação, já existente, de modo a permitir a busca de *feedback* após a construção do código;
- especificar e implementar o ambiente do professor, de modo a facilitar a avaliação do estudante;
- especificar e implementar um módulo adicional na plataforma *moodle*, com o subsistema de *feedback* colaborativo;
- realizar análise de viabilidade no subsistema por meio da oferta de um curso de programação;
- coletar dados do uso de ambiente para análise de usabilidade;
- implementar no ambiente melhorias com base nos métodos de usabilidade aplicado;
- reconstruir e reestruturar a interface do subsistema de *feedback* colaborativo, com base nos resultados alcançados;

## 1.3 METODOLOGIA DO TRABALHO

Após estudo detalhado dos sistemas existentes, a metodologia utilizada fundamentou-se na execução de uma pesquisa experimental, por meio do desenvolvimento de um subsistema dentro dos contextos de educação em computação, educação a distância, a utilização de uma Ambiente Virtual de Aprendizagem (AVA) e a utilização de um ambiente de programação. Foi construído um subsistema de *feedback* colaborativo com o propósito de auxiliar estudantes em programação a avaliarem soluções de outros estudantes.

O subsistema foi implementado na linguagem de programação chamada Hypertext Preprocessor (PHP) em uma plataforma de ensino de programação no AVA *Moodle*, no contexto de um ambiente chamado Laboratório de Ensino a Distância (LabEAD) da Universidade Federal do Pará.

Foi ofertado um Curso Básico de Programação para experimentar a viabilidade do subsistema. O curso teve carga horária de dez horas, divididas em dois módulos e totalmente a distância por meio do Ambiente de Programação integrado ao *Moodle*. Cada módulo apresentou um determinado teor, como, lista de exemplos, lista de exercícios e questionários de programação referentes a este conteúdo. Para coleta de dados, foram aplicados métodos de usabilidade para

identificar a capacidade de interatividade do subsistema com os usuários, os métodos aproveitados foram a Avaliação Heurística e o Ensaio de Interação. Logo, este curso fez parte de um estudo que verifica o quanto a utilização do subsistema é viável, quais aspectos podem ser aproveitados e quais as melhorias devem ser realizadas em trabalhos posteriores.

## 1.4 ORGANIZAÇÃO DO TEXTO

Além do capítulo introdutório, este trabalho está subdividido em quatro outros capítulos:

- Capítulo 2: aborda os desafios da aprendizagem de programação e suas principais dificuldades.
- Capítulo 3: descreve o subsistema de *feedback* colaborativo, focalizando no processo de geração de *feedback*.
- Capítulo 4: trata da análise de viabilidade e usabilidade do subsistema apresentando os resultados obtidos no experimento.
- Capítulo 5: tece as considerações finais, descreve trabalhos futuros e dificuldades encontradas durante o percurso da pesquisa.

# 2 QUESTÕES E DESAFIOS DA APRENDIZAGEM EM PROGRAMAÇÃO

Neste capítulo serão abordados os grandes desafios que o ensino de programação enfrenta, sobretudo no meio acadêmico. As dificuldades enfrentadas nas práticas de laboratórios, a sobrecarga que os professores possuem na mediação da aprendizagem, a falta de um *feedback* com qualidade e de que forma caminhar para o desenvolvimento de habilidades na autoavaliação da aprendizagem de conceitos introdutórios de algoritmo e programação.

## 2.1 DESAFIOS PARA O ENSINO DE PROGRAMAÇÃO

O ensino de programação, em geral, pressupõe um grande empenho por parte do estudante, sendo, desta forma desafiadora. Para Castro et al. (2004), a dificuldade começa para os que ingressam em turmas heterogêneas, no qual, alguns estudantes são provenientes de cursos técnicos e outros já são programadores que buscam na educação superior uma formação.

Além desta, outro desafio diz respeito ao acompanhamento por parte do professor que é prejudicado por problemas tais como o tempo reduzido na prática em laboratório e a sua sobrecarga de trabalho fazendo com que o estudante não tenha a oportunidade de analisar suas soluções e de formular suas dúvidas.

### 2.1.1 Práticas de laboratório

A aprendizagem, quando baseada em experimentação e construção, favorece o desenvolvimento de atitudes e destrezas cognitivas de alto nível intelectual (GALIAZZI et al., 2001). Portanto, a construção de programas de computadores, como uma atividade de experimentação, não pode ser vista apenas como a aplicação de conhecimentos para construção e execução de uma solução computacional, sem uma ampla discussão de seus resultados.

Assim como acontece em experimentos nas disciplinas de Física (BLÜMKE, 2002) ou Matemática (POLYA, 1978), a construção de uma solução, por meio de um algoritmo ou programa em linguagem de programação, é vista não apenas como a produção de um conjunto de resultados, mas como um momento de trabalho, reflexão, análise, questionamento, interpretação, troca de ideias, tomada de decisões e de conclusões. Mais do que uma atividade que estimula a curiosidade do estudante, o processo de construção de uma solução deve ser considerado uma importante prática pedagógica desenvolvida, em que o construir é importante, mas o refletir é primordial.

O processo de ensino-aprendizagem de programação envolve inúmeras dificuldades, relacionadas aos diferentes níveis de conhecimento dos estudantes. Segundo Tobar et al, (2001), um

fator agravante é o obstáculo encontrado pelos professores para acompanharem efetivamente as atividades laboratoriais de programação, devido ao grande número de estudantes que estão, geralmente, sob a supervisão de um professor. Sem esse acompanhamento, as atividades serão cumpridas sem a necessária reflexão da solução pelo estudante, sem analisar as tarefas realizadas ou não tenha conseguido formular suas dúvidas e obter suas respostas, perdendo-se a oportunidade de favorecer a análise, síntese e avaliação da experiência. Como consequência, soluções que poderiam ser bem elaboradas tendem a ser desprezadas como um "problema resolvido", ou seja, abandonam suas tarefas sem a reflexão esperada. Para Stamouli e Huggard (2006), é importante oportunizar uma revisão das decisões tomadas, pois, embora a compreensão da correção lógica do programa seja importante, a análise da solução pelo estudante afeta diretamente na construção das estratégias adotadas para solucionar problemas que serão enfrentados posteriormente.

### 2.1.2 Sobrecarga do professor na mediação da aprendizagem

Na avaliação das soluções dos estudantes em atividades de resolução de problemas com linguagens de programação ou algoritmos, além da correção do programa em si, o professor pode utilizar diferentes métodos para avaliar o desempenho do estudante. Assim como nos experimentos de Ciências, o professor pode (KNOX et al., 1996): fazer perguntas para o estudante sobre a sua solução de forma que este seja obrigado a descrevê-la, provocando a reflexão; fornecer novas atividades ou novos problemas baseados no problema resolvido e; ler o relato de um estudante sobre a caminhada de sua construção, avaliando sua conclusão sobre a solução encontrada ou ainda estimulando a avaliação dos pares.

A avaliação é importante porque pode estimular os estudantes a pensar sobre suas soluções e sobre o processo de resolução de problemas adotado. Portanto, requerer que os estudantes mantenham um relato sobre suas atividades é uma maneira de provocar a reflexão diante do problema em questão. Por outro lado, quando o professor acompanha um elevado número de estudantes, a mediação passa a ser um obstáculo. Nos casos em que são solicitadas "sínteses" ou "análises" explicativas de suas soluções, há uma sobrecarga de trabalho para o professor na avaliação das mesmas e uma consequente demora no *feedback* do professor para essas descrições. O *feedback* é considerado um importante elemento para aprendizagem, visto que permite ressaltar as dissonâncias entre resultado pretendido e o real, entretanto, um *feedback* pouco eficaz ou demorado prejudica a percepção de estudante quanto ao seu progresso, o que pode reduzir sua motivação (SCHUNK, 1989). Isso tende a acontecer quando o processo de avaliação e o *feedback* são centradas apenas no professor.

### 2.1.3 O *feedback* como um processo de transmissão de informações

Segundo Nicol e Macfarlane-Dick (2006), um maior comprometimento e responsabilidade do estudante no processo de ensino-aprendizagem implica em mudanças na forma de conduzir o processo de avaliação e *feedback*. Em uma abordagem de ensino-aprendizagem centrada no estudante, o *feedback* não pode ser concebido como um processo de transmissão, no qual os professores "transmitem" informações que orientam e conduzem a correção e a melhoria dos trabalhos acadêmicos.

A concepção do *feedback*, como um processo de "transmissão de informações", frequentemente centrado no professor, tem sido contestada por diversos pesquisadores (YORKE, 2003; BOUD, 2000; SADLER, 1998; NICOL e MACFARLANE-DICK, 2006). Alguns dos principais argumentos são relacionados a seguir:

(1) Existem fortes indícios de que as mensagens de *feedback* sejam complexas e difíceis de se compreender (LEA e STREET, 1998), portanto, os estudantes necessitam de oportunidades, por exemplo por meio de discussão, para compreender o *feedback* fornecido e utilizá-lo para regular seu desempenho (HIGGINS, HARTLEY e SKELTON, 2001)(IVANIC, CLARK e RIMMERSHAW, 2000).

(2) O *feedback* fornecido influencia na forma como os estudantes aprendem (DWECK, 1999) ou como constroem suas próprias estratégias e desenvolvem habilidades de autoavaliação de aprendizagem (ZIMMERMAN, 2000)(NICOL, 2007). Nesse sentido, o *feedback* deve ser fornecido não apenas direcionado para a correção de uma solução específica, mas considerando as estratégias adotadas pelo estudante ao longo de diversas soluções, visando contribuir para o aperfeiçoamento dessas estratégias.

(3) O *feedback*, como um processo cognitivo que envolve apenas a "transferência de informações", é uma forma de ignorar a relação que possui com a motivação do estudante (NICOL e MACFARLANE-DICK, 2006) (SCHUNK, 1989), já que o *feedback* de pares, professores e monitores influencia nas crenças de autoeficácia do estudante. O *feedback* é considerado um elemento essencial para aprendizagem, porque concede sobressair as dissonâncias entre o resultado pretendido e o real.

(4) A concepção de "transmissão de *feedback"* concentra no professor o esforço para produzir esse *feedback*, o que gera uma sobrecarga de trabalho que aumenta, conforme o número de estudantes e turmas, o que nem sempre garante a eficácia de seu apoio no processo de ensino-aprendizagem.

(5) O *feedback* quando não induz a uma reinterpretação, por parte do estudante, das causas que provocaram falhas em suas soluções, deixa de contribuir para a construção de habilidades de autoavaliação (NICOL e MACFARLANE-DICK, 2006), necessárias para prepará-los para aprender fora da universidade e ao longo da vida (BOUD, 2000).

## 2.2 EM DIREÇÃO A UMA APRENDIZAGEM DE PROGRAMAÇÃO MAIS ATIVA

Ao procurar favorecer o desenvolvimento de competências e habilidades de autoavaliação na aprendizagem de conceitos introdutórios de algoritmo e de programação, esta pesquisa tem como pressupostos principais o engajamento ativo e a responsabilidade do estudante na gestão da aprendizagem (LEA, STEPHENSON e TROY, 2003). De acordo com Santos (2002), a avaliação entre pares é um processo de regulação que oferece potencialidades e reconhece a interação social como um recurso fundamental na construção do conhecimento, visto que os estudantes são colocados em situações de confronto, de troca e de decisão, e também são forçados a argumentar, expor ideias, planejar, dividir o trabalho, entre outras circunstâncias. Para isso, as subseções seguintes descrevem os marcos conceituais que orientaram a adoção de um modelo de referência para o desenvolvimento das tecnologias de apoio ao aprendizado de algoritmos e programação.

### 2.2.1 A importância de uma abordagem reflexiva, cooperativa e colaborativa

Como forma de tornar a aprendizagem mais ativa, Shang et al. (2001) sugerem ampliar as experiências colaborativas, podendo-se criar pequenos grupos de estudantes, sugerindo questões ou colocando-os em situações de tomadas de decisão. Como exemplo de experiências colaborativas, Lazakidou e Retalis (2010) propõem o uso de estratégias de aprendizagem colaborativa apoiada por computador para auxiliar na aquisição de competências de autoavaliação na resolução de problemas de matemática.

Na aprendizagem de programação, após a resolução de um problema de programação, os estudantes podem ser convidados a descrever o trajeto realizado para chegar à solução e discutir, com seus pares, resultados e decisões adotadas. Por exemplo, na metodologia proposta por Menezes et al. (2008), adaptada de Polya (1978), os estudantes escrevem sua própria compreensão sobre um problema e o texto produzido pode sugerir uma pequena discussão em grupo, que pode ser valiosa mesmo para aqueles que não participaram do diálogo. Essa oportunidade, de relacionar a experiência com o diálogo, pode oferecer aos estudantes uma perspectiva nova sobre as suas convicções e valores, ajudando-os a construir os muitos possíveis significados para as suas experiências (SHANG et al., 2001).

Para Shang et al. (2001), as estratégias para ter significado à aprendizagem envolvem: dois tipos de diálogo, considerando o diálogo consigo mesmo (pensar reflexivamente) e com os seus pares (professores, estudantes, monitores ou especialistas); e dois tipos de experiência, considerando a experiência de construir e a de observar as construções de outros. Cada um dos quatro modos de aprender (pensar, dialogar com outros, observar e fazer), tem seu próprio valor e a combinação deles agrega valor ao processo de aprendizagem, tornando-o mais atrativo e significativo para os estudantes.

Por isso, a importância de deixar o estudante agir por si mesmo e substituir grande parte das aulas expositivas por oficinas nas quais ele possa experimentar, descobrir, criar soluções, discutir e sistematizar seu conhecimento.

### 2.2.2 Construção de habilidades pela resolução de problemas

O estudante em sua mente deve possuir uma ideia clara das características que envolvem o problema, os atributos e as regras. Resumindo, logo de início, é preciso compreender o processo de resolução do problema como um sistema transformável, ou seja, observar vários caminhos para alcançar a solução.

Para o desenvolvimento de um programa computacional, o estudante passa por um processo de resolução de problemas. De forma similar ao ensino de Ciências e Matemática, vários modelos de resolução de problemas têm sido propostos para apoiar o processo de ensino-aprendizagem de programação. O que esses modelos possuem em comum é a definição de fases ou passos do processo de resolução de problemas. Por exemplo, a partir de uma adaptação das ideias de Polya (1978), Menezes et al. (2008) sugerem as seguintes etapas na especificação do que denominaram de Aprendizagem de Programação Apoiada por Computador (CSPL – *Computer Supported Programming Learning*): (1) compreensão do problema; (2) planejamento; (3) desenvolvimento; (4) avaliação do processo e seus resultados e (5) socialização dos resultados.

Segundo Lazakidou e Retalis (2010), um dos aspectos comuns da maioria dos modelos que apoiam o ensino de disciplinas de resolução de problemas está na fase inicial: quando o estudante é orientado a identificar o objetivo do problema, ele também é orientado a ajustar suas ações, considerando as metas identificadas. Ou seja, o estudante constrói um modelo básico interno do problema e esse modelo deve ser confrontado com as metas identificadas. A etapa de compreensão do problema é o entendimento do problema para o qual há necessidade de se construir um programa computacional e onde se definem os dados de entrada e suas propriedades, resultados esperados e as relações entre os dados de entradas e os resultados esperados.

Após ter construído internamente a configuração do problema e tomado conhecimento de algumas regras colocadas no problema, o estudante planeja, ainda que mentalmente, o conjunto de instruções que passarão a ser aplicadas na solução do problema. Em CSPL na etapa de planejamento, Menezes et al. (2008), sugerem o uso de tecnologias de apoio, onde o estudante pode registrar uma lista organizada de casos de teste, incluindo os dados de entrada e as respectivas saídas esperadas.

A partir do planejamento de sua solução, o estudante segue para o processo de elaboração (construção) de algumas dessas instruções, como as de leitura e escrita de dados. É o processo de codificação e teste das soluções alternativas em uma linguagem de programação.

A avaliação das instruções, seja executando mentalmente ou utilizando um suporte computadorizado, deve resultar na seleção de instruções que são concebidas como consistentes com as metas. Essas instruções ou bloco de instruções são registrados na memória como padrões positivos e passarão a ser aplicadas na solução do problema atual e de outros problemas posteriormente enfrentados. Para resgatar da memória esses padrões, Lazakidou e Retalis (2010) sugerem que a mediação, ao longo de todo o processo de resolução de problemas, pode incluir estratégias meta-cognitivas, como apoio, que podem ser verbais ou não.

A avaliação, portanto, é um processo de reflexão sobre cada etapa do processo de construção de programas, de modo a consolidar os pontos fortes e corrigir os pontos fracos observados em cada uma das etapas. Assim, durante a avaliação ou revisão dos pares, os estudantes também podem retroceder nas instruções ou comandos manipulados, confrontando-se resultados (ainda que parciais) com os objetivos e as ações. A diversidade na compreensão das regras, de um estudante para outro, também vem a ser uma variável que contribui para promover as interações em um ambiente de cooperação. Esse processo pode derivar novas ações e modificações nas instruções iniciais da solução do estudante. Visualizadores e simuladores de programas e algoritmos podem contribuir para a percepção do estudante quanto às instruções ou bloco de instruções que são mais apropriadas para a solução do problema.

Dessa forma, enquanto constrói as instruções, o estudante desenvolve um raciocínio ordenado e sequencial especialmente adequado para o aprendizado de algoritmos. Sem a devida atenção às regras, os algoritmos não se desenvolvem. De acordo com Anderson (2000), analisando os aspectos cognitivos, os estudantes escolhem a maneira de agir que parece mais indicada com base em três modalidades esquemáticas de conhecimento que constituem uma aliança entre a percepção, a memória e o raciocínio: (1) proveniente da leitura que fazem do problema; (2) declarativo, arquivado em suas memórias; (3) procedural, relativo aos procedimentos, de acordo com regras estabelecidas.

Na proposta de Menezes et al. (2008), a socialização de resultados é considerada uma experiência importante para o estudante, pois oportuniza a reflexão sobre suas soluções, já que é um momento de revisão de pares, no qual o estudante emite e recebe *feedback*. Portanto, é fundamental oportunizar a explicação da solução desenvolvida. Também na resolução colaborativa de problemas (LAZAKIDOU; RETALIS, 2010), a objetividade e a clareza das ideias são testadas e vivenciadas.

Entretanto, apesar dos modelos citados incluírem um tipo de *feedback* em pares, o estudante deve sentir-se bem a vontade em um grupo, com liberdade de se expressar e discutir, mesmo que não esteja no nível de conhecimento com seus pares, isso proporciona uma maneira de raciocinar com qualidade.

### 2.2.3 O papel do *feedback* no processo de autoavaliação

Pesquisadores argumentam que estudantes com habilidades de autoavaliação da aprendizagem são mais persistentes, inventivos, confiantes e empreendedores (PINTRICH, 1995)(ZIMMERMAN e SCHUNK, 2001). Segundo Nicol e Macfarlane-Dick (2006), o desenvolvimento da autoavaliação pode ser facilitado pela estruturação de ambientes de aprendizagem, por meio de metaformação cognitiva, autocontrole e proporcionando oportunidades para praticar a autoavaliação (SCHUNK e ZIMMERMAN, 1994)( PINTRICH, 1995).

A preocupação, à autoavaliação da aprendizagem, está em fornecer as condições para que os estudantes se beneficiem do *feedback* em direção a construção de sua autonomia e autoavaliação. De fato, o *feedback* deve ser interpretado, (re)construído e internalizado pelo estudante para que possa ter uma influência significativa na aprendizagem (IVANIC, CLARK e RIMMERSHAW, 2000).

O *feedback* é tratado como uma forma de oferecer ao estudante os mecanismos necessários para compreender claramente (SADLER, 1989): (1) quais são os objetivos da disciplina, ou seja, o que é considerado um bom desempenho, internalizando uma meta de aprendizagem a ser alcançada; (2) como o desempenho atual se relaciona com o desempenho esperado (para isso, os estudantes devem ser capazes de comparar o desempenho alcançado e o esperado); e (3) como agir para reduzir ou eliminar a lacuna entre o desempenho atual e o esperado.

Para que os estudantes possam reduzir a lacuna entre os objetivos educacionais esperados e os objetivos alcançados, Sadler (1989) sugere que estes devam possuir "algumas das competências de avaliação do professor". Para alguns autores, como Yorke (2003) e Boud (2000), esta observação levou à conclusão de que, além de melhorar a qualidade das mensagens de *feedback*, é preciso investir esforços na construção de habilidades de autoavaliação. Os argumentos de Sadler (1989) explicam o fato de que estudantes em melhores condições de autoavaliar suas produções alcançam

significativos progressos, mesmo quando o *feedback* fornecido é insuficiente (NICOL e MACFARLANE-DICK, 2006).

As dificuldades encontradas no ensino inicial das disciplinas que possui a disciplina de programação como base, motiva a pesquisa deste trabalho para o desenvolvimento de um proposta de um subsistema de *feedback* colaborativo, promovendo a autoavaliação entre os estudantes, proporcionando a avaliação em pares, provocando entre eles o conflito de ideias promovendo assim, interação entre seu grupo.

# 3 PROPOSTA DO SUBSISTEMA DE *FEEDBACK* COLABORATIVO: FOCO NO PROCESSO DE GERAÇÃO DO *FEEDBACK*

Esta pesquisa dá continuidade a criação de um visualizador e simulador de programas, denominado *Javatool* (MOTA et al., 2008), cuja finalidade é realizar a animação de programas que auxiliam no desenvolvimento das habilidades no ensino de algoritmos e programação. O visualizador de programas permite demonstrar detalhes de cada caminho da execução, por meio de demonstrações gráficas em 2D, fazendo com que o estudante observe e compreenda melhor as estruturas básica dos programas durante a execução do código.

Para agilizar as tarefas de avaliação deste ambiente, foram criadas técnicas de avaliação automatizadas para as soluções do estudante, por Moreira et al (2009), que permitem apreciação automática ou manual. Esse modelo adapta o uso de uma avaliação da complexidade do código, por meio da técnica estatística de Regressão Linear Múltipla com indicadores complexidades, aliada a um testador de código por entrada/saída.

O *Javatool* e o Avaliador Automático foram integrados ao Ambiente Virtual de Aprendizagem *Moodle* por Mota et al. (2009), devido à grande quantidade e de diferentes recursos que essa ambiente oferece. Essa integração permitiu que os dois subsistemas fossem vinculados a novos recursos que foram desenvolvidos especificamente para programação na plataforma *Moodle*: exemplos, tarefas e questionários de programação.

Este capítulo propõem um subsistemas de *feedback* para um Ambiente de Aprendizagem de Programação (O *JavaTool* e o Avaliador Automático), este ambiente serviu como base para concepção do mesmo, que também é abordado neste capítulo, e a experiência aplicada nesse ambiente com uma turma de Ciência de Computação, na qual serviu de estudo para a proposta da criação do subsistema de *feedback* incorporado neste mesmo ambiente de programação.

## 3.1 CONTEXTO DA PROPOSTA: AMBIENTE DE APRENDIZAGEM DE PROGRAMAÇÃO

Conforme já citado, a disciplina de programação possui dificuldades, sendo de extrema importância em um curso de computação. Entre as dificuldades enfrentadas, a assimilação de códigos pelos estudantes é uma delas. O desenvolvimento de ferramentas que facilitam o ensino de programação é um dos meios encontrados para auxiliar o estudante em seu percurso acadêmico. Diversas abordagens foram utilizadas para criação dessas ferramentas, entre elas, o ensino de programação relacionada a linha de visualização de programas no sistema de computação gráfica e da animação para esclarecer e expor programas de computador, processos e algoritmos. Os itens

seguintes, explicam os subsistemas que fazem parte do Ambiente de Programação implementado no Moodle.

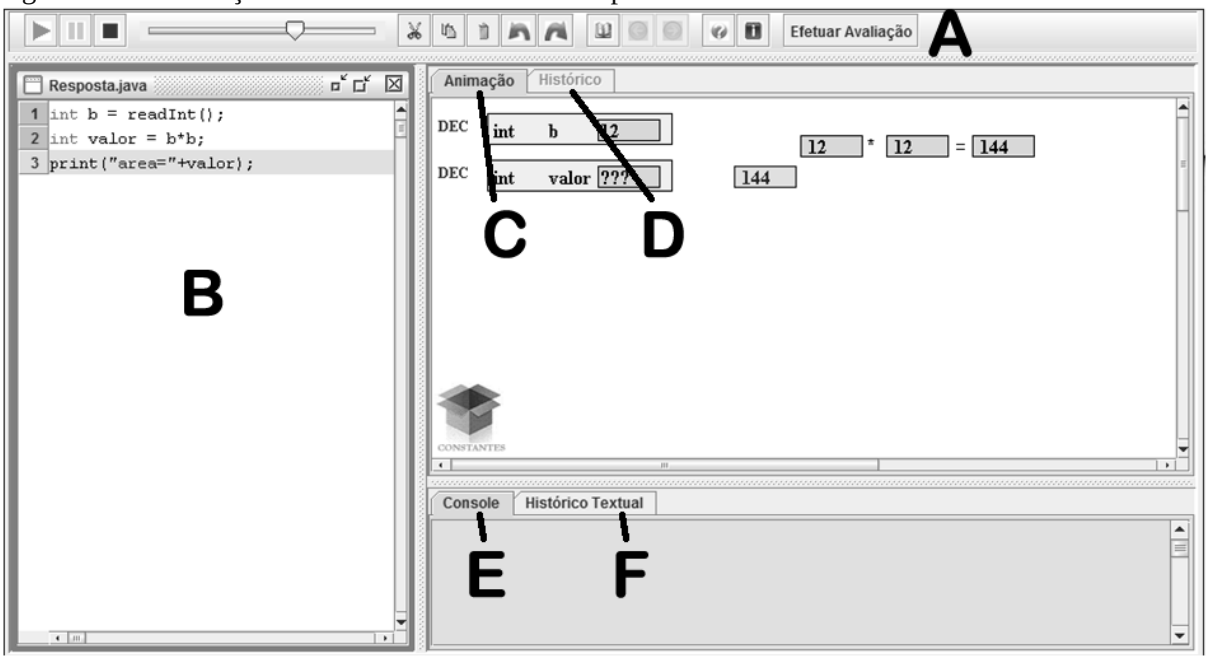
### 3.1.1 Simulador e visualizador de programas *Javatool*

Fundamentado nos conceitos de animação de códigos, foi desenvolvido uma ferramenta para favorecer o ensino de programação, denominado *Javatool* (MOTA et al., 2008). Entre suas caracteristicas principais está a visualização e animação de código *Java,* a partir de seu código fonte editado pelo usuário no ambiente.

O *Javatool* (MOTA et al., 2009) possibilita a animação textual e gráfica do código mostrando o seu comportamento na execução, para que o estudante possa compreender corretamente a lógica do programa. Dentre os recursos da linguagem *Java* disponíveis estão: tipos primitivos, *arrays* unidimensionais, estruturas de seleção e repetição, alguns métodos da classe *Math* e a criação e chamada de métodos simplificados.

Na Figura 1, pode-se observar uma tela do funcionamento do simulador, integrado na plataforma *Moodle*, após a resolução de uma questão de programação.

Figura 1 - Visualização do ambiente do *Javatool* na plataforma *Moodle*




Fonte: (MOTA et al., 2009**).**

A área ‘A’, demarcada na Figura 1, representa a barra de ferramentas. Nesta, estão localizados o *menu* de animação e histórico, além dos botões de ajuda, informações e armazenamento do código escrito pelo estudante. O editor é representado pela área ‘B’, onde se observa o uso do recurso de cor nas palavras reservadas da linguagem. A área ‘C’ corresponde ao

painel de animação, onde as estruturas são criadas e podem ser visualizadas. A área ‘D’ apresenta o resultado da saída do algoritmo executado. As áreas ‘E’ e ‘F’ (desabilitadas durante a simulação) são responsáveis pela representação gráfica e textual das operações realizadas durante a simulação.

No decorrer de um curso de programação, é importante que o estudante tenha acesso a materiais didáticos básicos como conteúdos e exemplos. Para a plataforma *Moodle*, foi desenvolvido um novo tipo de recurso, denominado “exemplo de programação”. O professor cadastra os exemplos ou pode salvar uma solução desenvolvida pelo estudante na base de exemplos. Os exemplos estão disponíveis como conteúdos e podem ser utilizados de acordo com os critérios do professor ou por livre exploração do estudante. Todos os exemplos podem ser visualizados com animação de código.

Também foram desenvolvidos dois novos tipos de atividades: “tarefa de programação” e “questionário de programação”. A tarefa de programação é um problema, cuja solução é executada no ambiente de simulação. O questionário contém questões diversas, selecionadas pelo professor, sendo que as questões de programação também são executadas no ambiente de simulação. O estudante pode consultar as questões e o histórico de suas tentativas com as respectivas notas (Figura 2), que podem ter sido geradas pelo avaliador automatizado ou pelo professor.

Figura 2 - Visualização inicial no questionário de programação

| Pergunta | Texto da pergunta | Histórico das respostas | Melhor nota |
|---|---|---|---|
| 1 | Criar um programa que leia 2 números do tipo double. Escreva no console o menor valor entre esses números, use print(menorvalor). Se os valores foram iguais, use print("iguais"). | #1(9.7), #2(0.0), #3(0.0), #4(0.0), #5(0.0), #6(0.0), #7(0.0), #8(0.0), #9(0.0), #10(0.0), #11(0.0), #12(0.0), #13(0.0), #14(9.7), #15(9.7), #16(9.7), #17(0.0), #18(9.5), #19(0.0), #20(0.0), #21(0.0), #22(9.5), #23(9.4), #24(9.4), #25(9.4), #26(9.7), #27(0.0), | 9.7 |
| 2 | Ler um número de um a sete do tipo inteiro, correspondendo ao dia da semana. Fazer um programa para escrever o dia ("domingo" caso o numero lido for 1, "segunda" caso o numero lido por 2 e assim por diante ). Quando o usuario digitar um valor que não estiver entre 1 e 7 faça print("dia invalido"); obs.: os dias NÃO devem conter acentos, cedilhas(s) e devem ter todas as letras minusculas. | #1(8.6), #2(8.6), #3(9.2), #4(0.0), #5(0.0), #6(0.0), #7(8.6), #8(9.2), #9(8.7), #10(9.2), #11(8.6), #12(8.6), #13(6.9), #14(9.2), | 9.2 |
| 3 | Fazer um programa para mostrar os números inteiros de 1 a n (inclusive o numero 1 e o numero n) em ordem **crescente**. n é um numero inteiro inserido pelo usuário. Use print(numero); | #1(9.9), #2(9.9), #3(9.9), #4(9.9), #5(9.9), | 9.9 |
| 4 | Fazer um programa para mostrar os números inteiros de 1 a n (inclusive o numero 1 e o numero n) em ordem **decrescente**. n é um numero inteiro inserido pelo usuário. Use print(numero); | #1(9.9), #2(9.9), #3(9.9), | 9.9 |

Fonte: (MOTA et al., 2009**).**

Com essa consulta, o estudante possui a facilidade de administrar todo seu percurso em um questionário de programação dentro do ambiente.

### 3.1.2 Subsistema de avaliação automática

O subsistema de avaliação permite a avaliação automática ou manual, com lançamento de notas e comentários. O modelo de avaliação automática (MOREIRA et al., 2009) combina o uso de

uma avaliação da complexidade do código, por meio da técnica estatística de Regressão Linear Múltipla, com indicadores de complexidade, aliada a um testador de código por entrada/saída. O simulador é responsável por acionar a avaliação automática.

Os indicadores utilizados pelo avaliador de código são métricas da Engenharia de Software. A técnica consiste em extrair da solução do estudante o valor obtido com a aplicação de cada uma das métricas descritas por Moreira et al.(2009). Esses valores são submetidos ao modelo de Regressão Linear Múltipla, que consiste em uma equação linear obtida previamente através de treinamento em uma base de testes. Neste caso, as métricas são as entradas da fórmula que calcula diretamente a nota do estudante. Em seguida, é verificado se a solução retorna o resultado esperado para as entradas previamente informadas.

Para o funcionamento correto do avaliador, o professor deve cadastrar uma solução considerada "resposta-modelo" para o problema. Assim, se não houver erro de compilação, o sistema avalia e obtém um valor para a nota do estudante, que pode ser modificada dependendo do resultado da avaliação por testes de entrada/saída. O sistema mantém a maior nota obtida em cada questão para calcular a nota final do questionário, o estudante também pode consultar o histórico de tentativas que realizou em cada questão, em interface similar ao professor (Figura 3).

Figura 3 - Interface com o histórico de resposta e a melhor nota

| Pergunta | Texto da pergunta | Histórico das respostas | Melhor nota |
|---|---|---|---|
| 526 | Imprima no console "tratamento especial" se o cliente é regular e "tratamento normal" se o cliente não é regular. Clientes regulares compram mais de R$ 1000 por ano **ou** são clientes a mais de 20 anos. Receba o valor das compras por ano e o número de anos que o usuário é cliente como entradas do programa (nesta ordem). | #1(9.7), #2(9.7), #3(9.7), | 9.7 |
| 527 | Fazer um programa para ler 2 notas, calcular a média aritmética. Exibir no console a mensagem "reprovado" se media inferior a 5.0, "recuperacao" se media entre 5.0 e 6.9, "aprovado" se media maior ou igual a 7.0. | #1(9.6), #2(0.0), | 9.6 |
| 528 | Fazer um programa para mostrar os números pares inteiros de 1 a n (inclusive) em ordem crescente. N é um valor inteiro inserido pelo usuário. | #1(9.9), | 9.9 |
| 529 | Fazer um programa para mostrar os números impares inteiros de 1 a n (inclusive) em ordem crescente. N é um valor inteiro inserido pelo usuário. | #1(9.9), | 9.9 |
| Nome | Aluno 1 | Nota | 9.8 |

Fonte: (MOREIRA et al., 2009).

Com esse histórico de resposta e a melhor nota, o estudante procura facilitar o entendimento no seu percurso em um questionário de programação dentro do ambiente.

### 3.1.3 Avaliação do ambiente de aprendizagem de programação a partir de uma experiência com turmas de Sistema de Informação

Após a implementação e integração do avaliador de código no *Javatool* e do ambiente de programação na plataforma *Moodle*, um experimento foi realizado no ambiente com os estudantes de duas turmas de programação, uma localizada na cidade de Belém (PA) e outra na cidade de Marabá (PA). Nessa experiência, dois aspectos importantes foram considerados na avaliação

realizada por Brito et al. (2011): usabilidade e eficácia educacional. Este item sintetiza as principais conclusões a que chegaram este experimento de uso do ambiente.

A avaliação da usabilidade teve como referência os estudos de Kulyk et al. (2007), o quais afirmam que está deve ser parte integrante do processo de um projeto e implementação sendo realizada em fases. Assim, a avaliação do experimento possuiu três fases: (1) os primeiros protótipos foram avaliados com inspeções heurísticas e o resultado dessas avaliações foram usados para melhorar o sistema; (2) a versão melhorada do sistema foi avaliada com técnicas de consulta, experimentos controlados e estudos observacionais, sendo que os resultados da avaliação ajudaram a corrigir erros e melhorar o sistema e finalmente, (3) estudos observacionais foram usados para avaliar o uso do ambiente em cenários reais de aprendizagem em laboratório.

Segundo Brito et al. (2011), a diversificação nas estratégias de avaliação da usabilidade é importante porque nos estudos observacionais, assim como nos questionários, o ambiente e as tarefas executadas pelos estudantes são parcialmente controlados pelos avaliadores. Assim ao longo da evolução do projeto, do ponto de vista da usabilidade, foram considerados:

- avaliações informais, em que a opinião dos estudantes foi capturada após o uso do sistema (MOTA et al. 2009);

- avaliações heurísticas, realizadas pelo professor especialista, sobre o uso dos recursos interativos e as características específicas para sistemas de visualização e simulação (MOTA et al. 2008) (MOREIRA et al. 2009) (MOTA et al. 2009);

- estudos observacionais, em que os projetistas e avaliadores atentam para o uso do sistema e registraram aspectos importantes sobre este, verificando pontos a melhorar ou a especificação de novos requisitos (MOTA et al. 2009);

- questionários, por meio dos quais os estudantes foram consultados sobre diferentes aspectos do sistema.

A pesquisa passou por mais de uma fase em suas avaliações de usabilidade, embora o uso contínuo em disciplinas de programação favoreça a realização de experimentos controlados e não controlados de seu uso. Nos experimentos controlados, os estudantes, as opiniões e as tarefas executadas possuem um controle, na qual podem fornecer informações sobre a eficácia, a facilidade de uso, eficiência e outros aspectos relevantes que impactam na usabilidade do sistema. Mesmo nos experimentos não controlados, o registro das ações dos estudantes são mantidas para posterior comparação com de desempenho e usabilidade.

Quanto à eficácia educativa, Brito et al.,(2011) na tentativa de comprovar as hipóteses de Hundhausen et al. (2002), de que o esforço dedicado pelos estudantes às tarefas relacionadas com a

visualização é mais importante que o conteúdo das visualizações do *Javatool*. Além da visão passiva de visualizadores de programas e algoritmos, Naps et al. (2003) desenvolveram uma taxonomia em níveis das diferentes formas de interagir com essas tecnologias. Nessa taxonomia, os autores sugerem uma estrutura hierárquica, aonde um alto nível de envolvimento do estudante conduz a maiores benefícios educacionais (Quadro 1). Considerada em nível crescente, cada nível, a partir da Visualização, inclui todos os níveis anteriores.

Quadro 1 - Taxonomia para nível de envolvimento com visualizadores de programas e algoritmos

| Nível | Descrição |
|---|---|
| Nível 1: Visualizar (V) | Considerado como o núcleo formador do envolvimento do estudante, considera que um estudante pode visualizar uma animação passivamente, mas também pode exercer controle sobre a direção e o ritmo da animação, usando diferentes janelas (cada uma apresentando uma visão diferente) ou a utilização de acompanhamento com explicações textuais ou fonéticas. |
| Nível 2: Responder (R) | A atividade principal desta categoria é responder a questões quanto às visualizações apresentadas pelo sistema. O estudante utiliza a visualização como recurso para responder a perguntas. |
| Nível 3: Modificar (M) | Implica em modificar a visualização. O exemplo mais comum dessa modificação é permitir que o estudante possa alterar a entrada do algoritmo a fim de explorar o comportamento do algoritmo em diferentes casos. |
| Nível 4: Construir (C) | Os estudantes constroem suas próprias visualizações dos algoritmos em estudo. Hundhausen e Douglas [2002] identificaram duas formas principais em que os estudantes podem construir visualizações: geração direta e construção manual. É importante notar que a construção não significa necessariamente codificar o algoritmo. |
| Nível 5: Apresentar (A) | Implica em apresentar a visualização para uma plateia, promovendo discussão e *feedback* entre os pares. |

Fonte: (NAPS et al., 2003).

Uma nova avaliação foi realizada, utilizando a taxonomia de Naps et al. (2003). Nessa avaliação, e em comparação com outros ambientes, o ambiente proposto foi classificado no nível quatro, incluindo todas as categorias: Visualizar, Responder, Modificar e Construir. O nível cinco pode ser alcançado a partir da adoção de uma metodologia na qual o estudante apresenta a visualização para a turma, registrando a sua solução, postando e comentando nos fóruns ou outros mecanismos de interação disponíveis. Foram realizados estudos observacionais e aplicação de questionários nas turmas de Belém e Marabá (29 e 47 estudantes respectivamente) da disciplina de algoritmos e programação do curso de Sistema de informação. Os questionários foram aplicados no começo do curso, com a finalidade de capturar os conhecimentos iniciais dos estudantes quanto ao desenvolvimento da lógica e capacidade de organização para problemas estruturados, e ao final do curso, com o intuito de avaliar a motivação quanto ao uso do ambiente de aprendizagem e a interface das tecnologias utilizadas.

De fato, por meio da interface do professor, especificamente desenvolvida para acompanhar as tentativas dos estudantes em cada questão (Figura 4), foi possível observar que, mesmo nas situações em que a solução resolve o problema (notas acima de sete), pelo menos 70% das soluções passam pelo processo de "refinamento". O uso do subsistema de avaliação incentiva o estudante a melhorar suas soluções e a refletir sobre o que considera um "programa correto". Além disso, o *feedback* automático reduz o tempo de espera do estudante pela avaliação de sua tarefa, ao mesmo tempo em que minimiza a sobrecarga do professor com as atividades de mediação.

Figura 4 - Interface do professor (avaliação por questão) para acompanhamento dos questionários de programação

| Pergunta | Texto da pergunta | Nome completo | Histórico das respostas | Melhor nota |
|---|---|---|---|---|
| 1 | Criar um programa que leia 2 números do tipo double. Escreva no console o menor valor entre esses números, use print(menorvalor). Se os valores foram iguais, use print("iguais"). | | | |
| | | Aluno 1 | #1(9.7), #2(0.0), #3(0.0), #4(0.0), #5(0.0), #6(0.0), #7(0.0), #8(0.0), #9(0.0), #10(0.0), #11(0.0), #12(0.0), #13(0.0), #14(9.7), #15(9.7), #16(9.7), #17(0.0), #18(9.5), #19(0.0), #20(0.0), #21(0.0), #22(9.5), #23(9.4), #24(9.4), #25(9.4), #26(9.7), #27(0.0), | 9.7 |
| | | Aluno 2 | #1(4.8), #2(4.6), #3(0.0), #4(0.0), #5(0.0), #6(0.0), #7(3.1), #8(3.1), #9(9.4), | 9.4 |
| | | Aluno 3 | #1(6.4), #2(9.5), | 9.5 |

| Pergunta | Texto da pergunta | Nome completo | Histórico das respostas | Melhor nota |
|---|---|---|---|---|
| 2 | Ler um número de um a sete do tipo inteiro, correspondendo ao dia da semana. Fazer um programa para escrever o dia ("domingo" caso o numero lido for 1, "segunda" caso o numero lido por 2 e assim por diante ). Quando o usuario digitar um valor que não estiver entre 1 e 7 faça print("dia invalido"); obs.: os dias NÃO devem conter acentos, cedilhas(s) e devem ter todas as letras minusculas. | | | |
| | | Aluno 1 | #1(8.6), #2(8.6), #3(9.2), #4(0.0), #5(0.0), #6(0.0), #7(8.6), #8(9.2), #9(8.7), #10(9.2), #11(8.6), #12(8.6), #13(6.9), #14(9.2), | 9.2 |

Fonte: (Brito et al., 2011**).**

Os resultados dos estudos observacionais revelaram que, pelo menos 40% dos estudantes utilizaram o recurso de "acelerar" as visualizações em uma tentativa de "passar mais rápido pela visualização" para receber resposta (*feedback*) do avaliador automático. Em resposta aos questionários, 95% dos estudantes declararam que o avaliador de código é útil para fornecer um *feedback* mais rápido para soluções incompletas ou não adequadas ao problema, sendo que 49% dos estudantes alegaram que o visualizador se apresentou útil para "rastrear" a execução nessas situações. Esses resultados impulsionaram a pesquisa (BRITO et al., 2011) em direção ao potencial do *feedback* e das possibilidades de ampliação desse por meio do *feedback* colaborativo, no qual o estudante não apenas recebe o *feedback*, mas também o fornece.

## 3.2 PROPOSTA DO SUBSISTEMA DE *FEEDBACK* COLABORATIVO

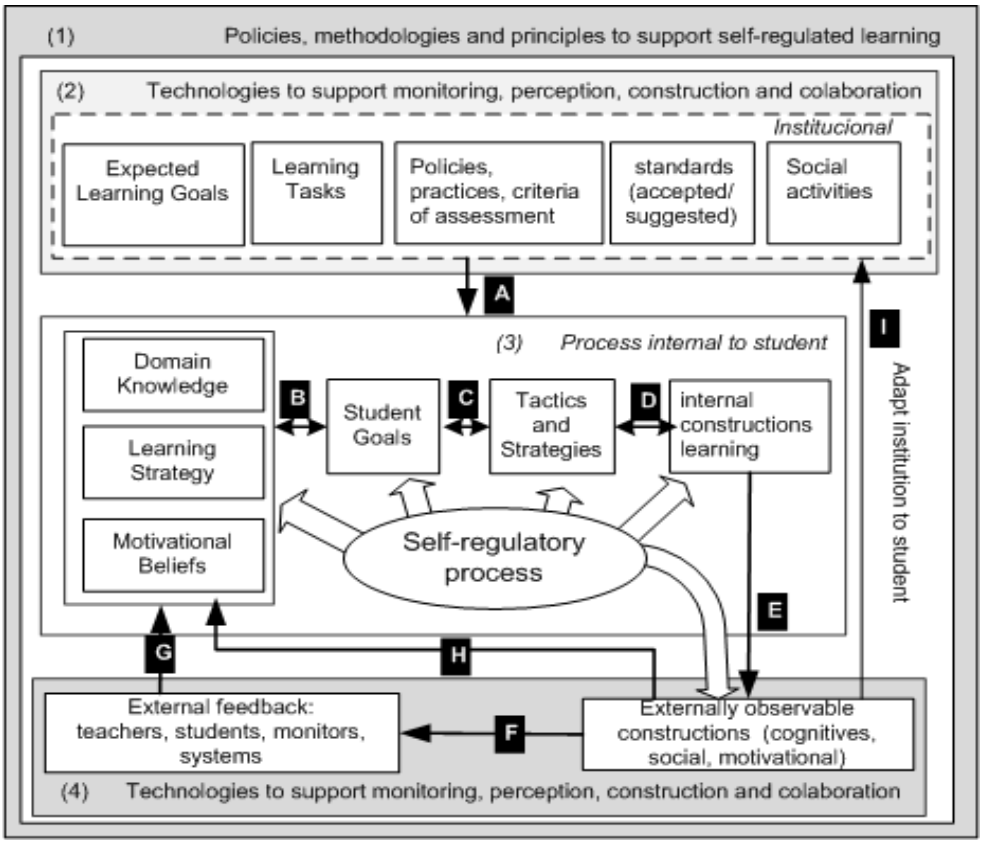
O ambiente de programação integrado ao *Moodle* permitiu utilizar tecnologias de interação já existentes na plataforma. Entretanto, com a finalidade de exercitar e desenvolver habilidades de autoavaliação, foi implementado uma nova funcionalidade, denominada de "Subsistema de *Feedback* Colaborativo", um subsistema de colaboração em que os estudantes podem fornecer *feedback* para as soluções de seus pares, de forma que o *feedback* não fique centrado no ambiente

de aprendizagem ou no professor. Assim, os estudantes exercitam e recebem *feedback* de outros estudantes baseado nos seus conhecimentos.

### 3.2.1 Modelo de referência para desenvolvimento e uso das tecnologias de apoio

Com base nos pressupostos mencionados no experimento realizado por Brito et al. (2011), estudos caminharam para um modelo conceitual (Figura 5), adaptado de Nicol e Macfarlane-Dick (2006) e Nicol (2007), com o objetivo conceber uma nova proposta de orientar o desenvolvimento e o uso das tecnologias de apoio à aprendizagem de programação. As adaptações no modelo de Nicol e Macfarlane-Dick (2006) foram realizadas para: (a) estabelecer a realimentação do processo com a finalidade de alçar o nível de envolvimento do estudante com decisões e práticas institucionais e adaptar os métodos de ensino às necessidades dos estudantes (NICOL, 2007); e (b) valorizar o desenvolvimento de autoavaliação, por meio da observação e do diálogo acerca das soluções construídas por si e por seus pares e destacar, no modelo, a importância das tecnologias de apoio que contribuam para a percepção do estudante quanto aos objetivos e seu desempenho.

Figura 5 - Modelo conceitual de autorregulação da aprendizagem




Fonte: adaptado de Nicol e Macfarlane-Dick (2006) e Nicol (2007).

O modelo está tracejado em princípios, metodologias e políticas de apoio à autorregulação[1] da aprendizagem (área 1), que propõe nortear, não apenas a construção e o uso de tecnologias de apoio mas também a adoção de abordagens e arquiteturas pedagógicas. Entender que as metodologias adotadas possam ser pautadas em princípios, tais como aqueles que auxiliam a construção de crenças de autoeficácia no estudante e as políticas institucionais devem apoiar essas práticas.

A área 2, originalmente definida por Butler e Winne (1995), e utilizada no modelo de Nicol e Macfarlane-Dick (2006), como as tarefas designadas pelo professor, foi expandida para abranger o apoio tecnológico à definição de objetivos esperados, políticas, práticas, critérios e padrões de avaliação, além de atividades que devem provocar experiências sociais relevantes para aumentar o nível de participação do estudante na instituição (NICOL, 2007). Assumir que parte dessas experiências sociais podem ser promovidas institucionalmente.

Adaptado a partir do modelo de Nicol e Macfarlane-Dick (2006), o processo interno ao estudante (área 3) procura indicar como o estudante monitora e regula sua aprendizagem e desempenho por meio de um tipo de "*feedback* interno". A adaptação do modelo foi realizada para reunir as tecnologias facilitadoras e destacar duas importantes contribuições do *feedback* (área 4): recebido a partir dos pares e produzido pelo estudante, constituindo assim uma das construções de colaboração; e, capturado, de forma implícita ou explicita, a partir de resultados externamente observáveis, como uma forma de retroalimentar o processo (planejamento de objetivos, tarefas, atividades sociais, práticas de avaliação) e garantir um dos princípios propostos por Nicol (2007) que é a adaptação da instituição às necessidades dos estudantes.

A autoavaliação do estudante (centro da Figura 5), assim como no modelo de Nicol (2007), é iniciada pelas atividades didáticas, planejadas a partir dos objetivos educacionais, políticas, critérios e padrões desejáveis. Com o apoio de tecnologias, o estudante constrói um entendimento próprio de objetivos e metas e determina seu planejamento (estratégias e táticas) para alcançá-los (A). O comprometimento do estudante com a tarefa proposta está fundamentado em seus conhecimentos preliminares, estratégias de aprendizagem e motivação (B). Com base em uma concepção interna dos objetivos e das tarefas a serem realizadas (C), o estudante projeta as estratégias para alcançá-las (D). As maneiras utilizadas em sua estratégia produzem resultados, alguns dos quais podem ser externamente observáveis (E), que são soluções de programação do estudante (parciais ou não) ou resultados de tarefas de colaboração.

---

[1] A palavra autorregulação mencionando no modelo conceitual, será adaptada para autoavaliação no decorrer do texto.

O *feedback* externo é gerado a partir das soluções apresentadas pelos estudantes (F) e pode ser originado por professores, monitores, outros estudantes ou pelo ambiente de aprendizagem (G). A autoavaliação é apoiada pela observação dos resultados alcançados (resultados externos) (H) e pela observação das soluções de outros estudantes, além do *feedback* externo, que pode ser produzido pelo ambiente de apoio a programação ou pelos outros participantes do processo, além das intervenções do professor, independente de resultados, a fim de elevar a autoeficácia do estudante naquela determinada tarefa.

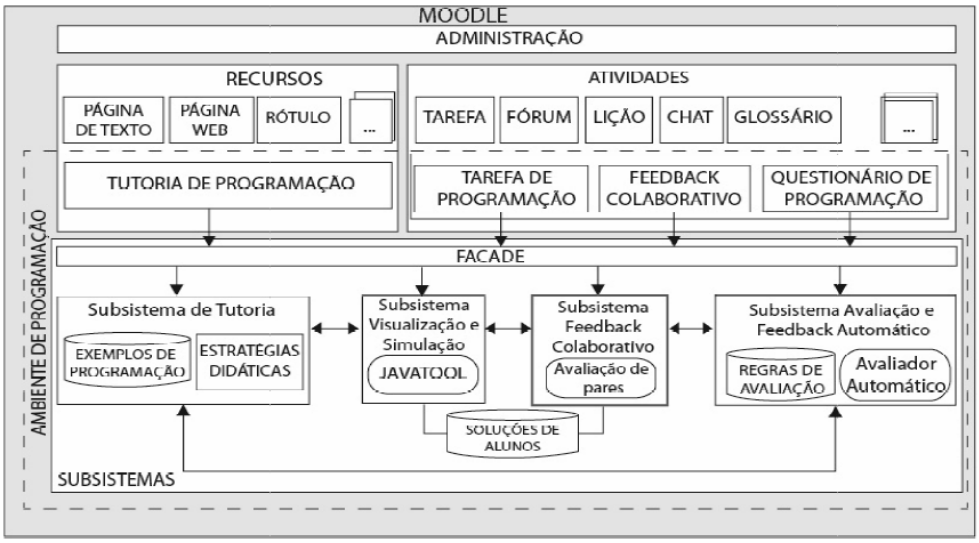
Resultados internos, para Nicol e Macfarlane-Dick (2006), referem-se às mudanças nos estados cognitivo, afetivos ou motivacionais que ocorrem durante o envolvimento com a tarefa. As interações com a tarefa geram *feedback* interno em diferentes níveis (cognitivo, motivacional, comportamental), portanto os resultados observáveis referem-se a diferentes tipos de construções (colaborativas, cognitivas e motivacionais). Nesse processo, o estudante realiza comparações e confronta os resultados alcançados e objetivos esperados para determinar se as estratégias adotadas são eficazes. As tecnologias de apoio, portanto, podem fornecer elementos que contribuam para a percepção do estudante nessas comparações, em direção ao desenvolvimento das competências de autoavaliação.

Para contribuir no processo de autoavaliação, o *feedback* fornecido pelo ambiente de apoio a programação, deve acontecer de forma interativa e no momento de construção da solução pelo estudante no sentido de contribuir para a motivação e percepção do estudante quanto ao seu progresso nas atividades realizadas.

### 3.2.2 Uma nova arquitetura para o ambiente de programação com a incorporação do subsistema de *feedback* colaborativo.

A nova arquitetura de integração foi construída a partir dos resultados de experimentos realizados por Brito et al. (2011) que apontam para um aumento no interesse dos estudantes no processo de refinamento da solução. Os resultados da pesquisa promoveram novas mudanças na arquitetura original de Mota et al (2009), agora com o objetivo de explorar os benefícios do *feedback* colaborativo no ambiente de programação já existente, colocar o estudante ativo no processo de autoavaliação das soluções, proporcionar o conflito de ideias e diminuir a sobrecarga de trabalho do professor (Figura 6).

Figura 6 - Arquitetura de integração do ambiente de programação ao *Moodle*, em destaque o subsistema




Fonte: SIROTHEAU et al (2012).

A Figura 6 demonstra a arquitetura do ambiente de programação integrado ao *moodle* por meio dos recursos (tutoria de programação) e das atividades (tarefa de programação, *feedback* colaborativo e questionário de programação) fazendo a comunicação através de uma fachada para os subsistemas do ambiente (tutoria, visualização e simulação, *feedback* colaborativo e avaliação e *feedback* automático).

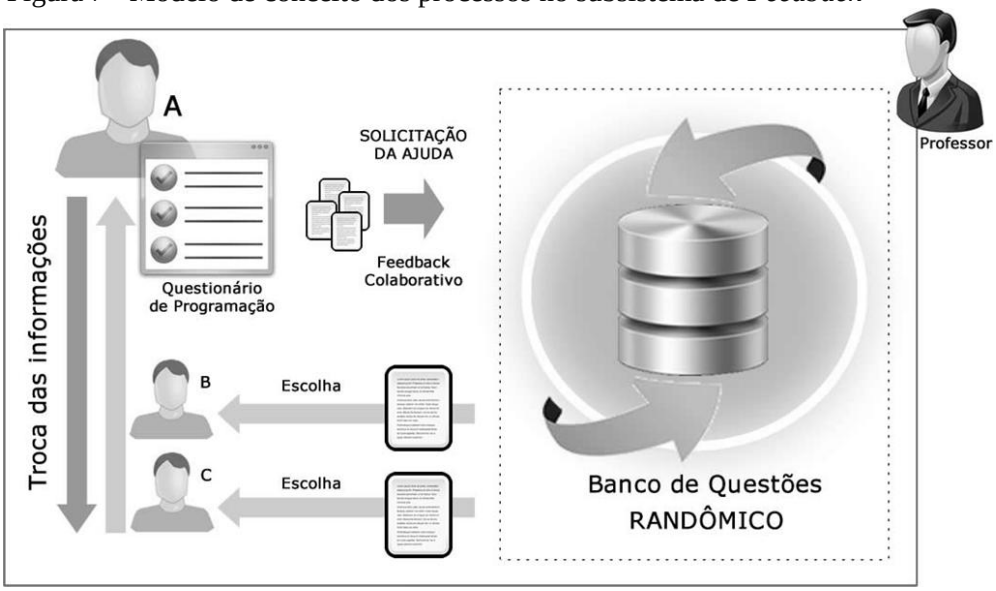
Para o professor, a arquitetura prevê um monitoramento (acompanhamento) do estudante por meio de um ambiente administrativo, na qual poderá acompanhar as contribuições, a evolução e o desempenho de cada estudante que o subsistema escolheu aleatoriamente para criação de seus pares. Além de facilitar o planejamento, um aspecto importante dessa nova arquitetura é que as informações fornecidas pelo professor não serão de intervenção, apenas de incentivos que deverão servir de apoio para facilitar aprendizagem do estudante de forma a fornecer informações adicionais, de modo a contribuir para a autoavaliação da aprendizagem do estudante.

Com base nos pressupostos teóricos da autoavaliação da aprendizagem, observar que a maior parte dos esforços para fornecer o *feedback* estavam concentrados no professor e no ambiente de aprendizagem, ou seja, o *feedback*, até então, foi concebido como um processo de transmissão da informação para orientar a correção e a melhoria das soluções do estudante. A partir dessa análise e do modelo adaptado, foram identificados requisitos que direcionaram para a construção de uma nova arquitetura integração para o ambiente.

## 3.3 DESCRIÇÃO DO SUBSISTEMA DE *FEEDBACK* COLABORATIVO

O subsistema de *feedback* colaborativo ou subsistema de autoavaliação é uma ferramenta de interação e de recursos para colaboração nos "questionários de programação" desenvolvidas no ambiente de aprendizagem de programação, com a finalidade de desenvolver habilidades de autoavaliação. A concepção central do subsistema é gerenciar as avaliações realizadas para soluções de outros estudantes e colocar o estudante de forma presente no processo de avaliação do banco de soluções do ambiente (Figura 7).

Figura 7 - Modelo de conceito dos processos no subsistema de *Feedback*




Fonte: Produção do próprio autor (2012).

A criação da ferramenta tem como meta promover a autoavaliação como forma de apoiar a compreensão do estudante quanto aos objetivos das disciplinas que tem a programação como base, proporcionando um desempenho qualitativo e esperado. Desta forma, a ferramenta possibilita ao estudante, desenvolver habilidades à medida que deixa de receber os conceitos prontos e acabados, aprende a buscar respostas e encontrar soluções para os problemas propostos com liberdade e responsabilidade, assim, desenvolvendo sua autonomia ao mesmo tempo em que regula suas metas, estratégias, cognição motivação e comportamentos.

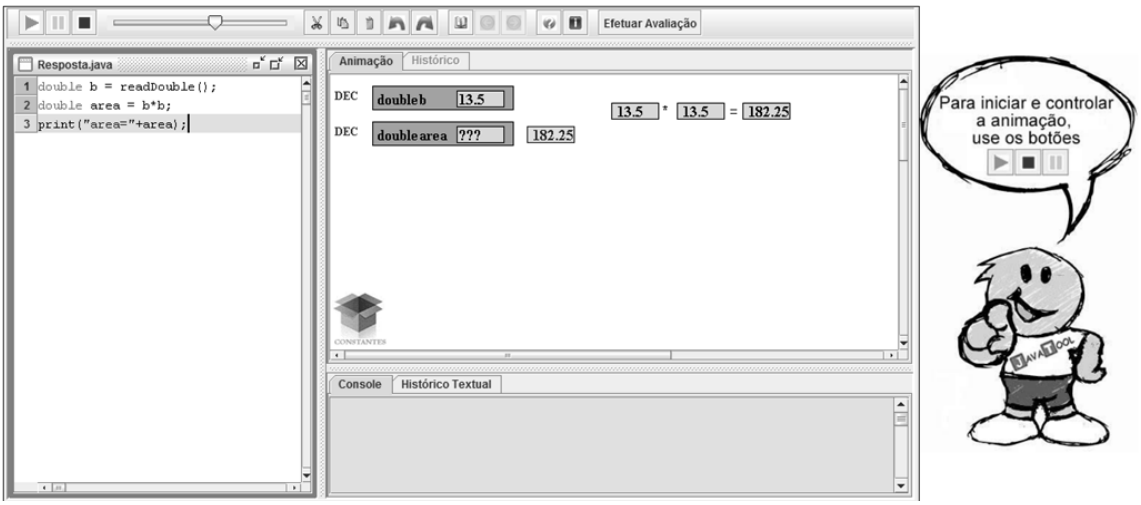
Por meio desse subsistema, os estudantes podem visualizar as soluções sugeridas pelos seus pares e contribuir para o desenvolvimento final das soluções de outros estudantes. O processo de seleção das soluções a serem avaliadas em pares é realizado aleatoriamente a partir da base de soluções de estudantes. Assim, o estudante recebe imediatamente uma informação sobre sua avaliação. Cabe ao estudante considerar relevante ou não o *feedback* fornecido.

Esses resultados relevam a importância do modelo de autoavaliação da aprendizagem na obtenção de resultados de excelência e maior engajamento do estudante, no qual não apenas recebe a contribuição do *feedback*, mas passa a ser ativo, sendo o centro do processo da avaliação, passando a fornecer o feedback. O desenvolvimento do subsistema levou em atenção todas as circunstâncias da pesquisa, de forma, que conclusões da pesquisa a que observou o aumento no interesse dos estudantes em melhorar suas soluções no ambiente de programação.

### 3.3.1 Funcionalidades do subsistema

Com base nessa concepção, foi proposto incorporar ao Ambiente de Ensino em Programação uma nova funcionalidade, com a finalidade de exercitar e desenvolver habilidades de autoavaliação. Com uma nova funcionalidade, o estudante é submetido a fornecer um *feedback* para dúvidas produzidas por outros, a escolha ocorre por meio de um processo randômico com varias condições (por exemplo, se o estudante já está em um processo de autoavalição, ele seleciona outro estudante), após passar por essas etapas, o subsistema agrupa em pares os estudantes, liberando o diálogo entre eles. Isso acontece, especificamente no questionário de programação, os estudantes procedem a atividade elaborada pelo professor escolhendo a questão para resolver, por meio da opção "responder", o ambiente *Moodle* executa o *Applet* com o *JavaTool* que concede ao estudante criar seu código, possibilitando ao estudante exploração completa do código partindo do principio que as maiores dificuldades do estudante em compreender programação esta nos passos do programa. No *JavaTool* o estudante cria/edita o programa, a interface tem como princípios recursos de animações que compila o código e depois exibe a animação no painel, contribuindo assim para o desenvolvimento dos estudantes em fases iniciais de programação (Figura 8).

Figura 8 - O *moodle* executando o *applet* com o *JavaTool*




Fonte: (MOTA et al., 2009)

Após os estudantes compilarem o seu código, na opção “Salvar alterações”, o ambiente armazena as informações no banco de dados e “chama” o *Web Service* requerendo a avaliação do programa escrito pelo estudante, o avaliador automático recebe a solicitação e calcula a nota com base na resposta modelo cadastrada pelo professor devolvendo assim para o *Web Service* o valor resultante da resposta que logo encaminha para o *Moodle* que salva e mostra a nota para o estudante. Esse processo ocorre quantas vezes o estudante solicitar dentro do questionário de programação, criando assim um registro histórico de notas e tentativas (Figura 9). Após a conclusão do questionário de programação, os estudantes que estiverem, por algum motivo, insatisfeitos com seu desempenho encontrados em seu histórico de respostas, poderá solicitar ajuda por meio da avaliação colaborativa ou são convidados a avaliar as soluções cadastradas por outros estudantes.

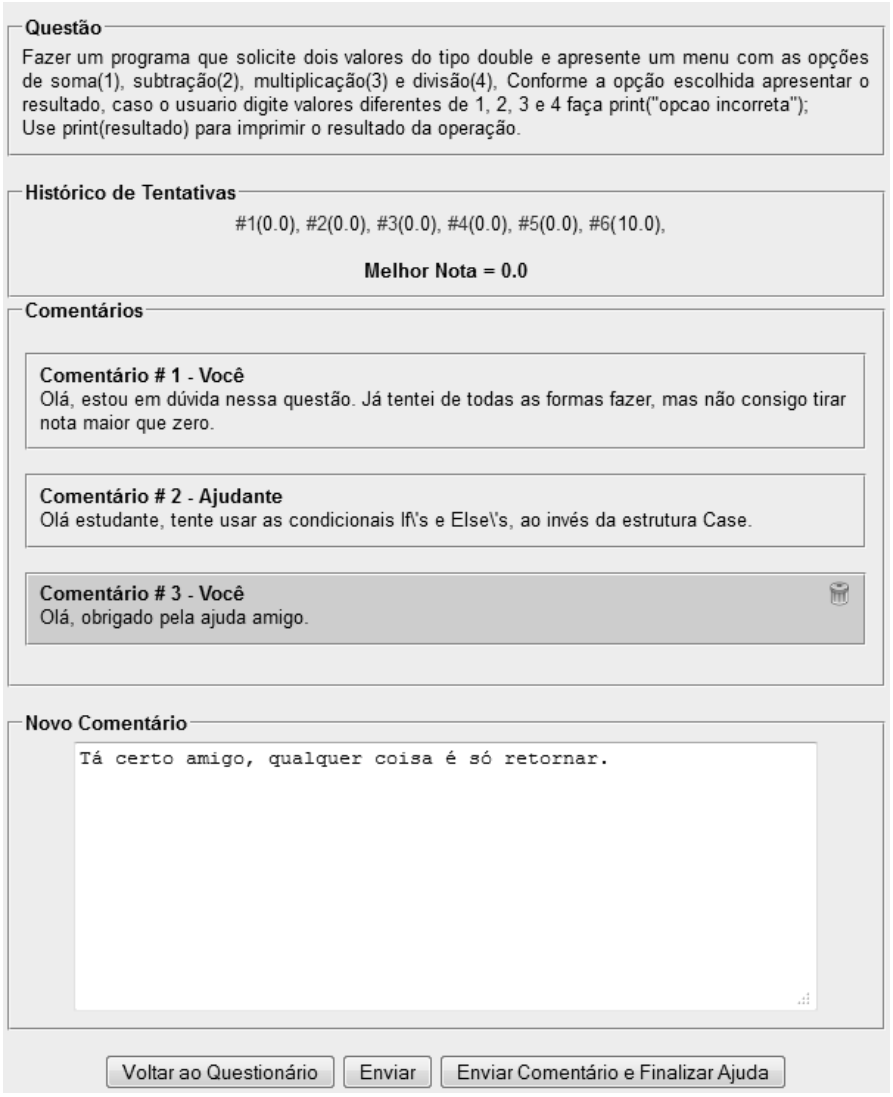
Figura 9 - Registro de histórico de tentativas no ambiente de programação

| Pergunta | Texto da pergunta | Histórico das respostas | Melhor nota | | Avaliação Colaborativa |
|---|---|---|---|---|---|
| 1 | Criar um programa que leia 2 números do tipo double. Escreva no console o menor valor entre esses números, use print(menorvalor). Se os valores foram iguais, use print("iguais"). | #1(9.7), #2(0.0), #3(0.0), #4(0.0), #5(0.0), #6(0.0), #7(0.0), #8(0.0), #9(0.0), #10(0.0), #11(0.0), #12(0.0), #13(0.0), #14(9.7), #15(9.7), #16(9.7), #17(0.0), #18(9.5), #19(0.0), #20(0.0), #21(0.0), #22(9.5), #23(9.4), #24(9.4), #25(9.4), #26(9.7), #27(0.0), | 9.7 | Responder | Pedir/Ver Ajuda |

Fonte: (Moreira et al., 2009).

Após expor o passo a passo da execução do componente de autoavaliação, a Figura 10 apresenta a *interface* do subsistema de *feedback* colaborativo incorporado ao ambiente virtual de aprendizagem. Na parte superior do ambiente do subsistema, o comando da questão no qual o estudante solicitou ajuda fica visível para o outro estudante que irá ajudar, facilitando assim o processo de formatação do *feedback*. Em seguida, o ambiente demonstra o histórico de tentativas do estudante que solicitou ajuda, junto com a melhor nota entre elas, dessa maneira, o estudante escolhido para contribuir, possui várias informações por meio desses históricos, auxiliando o estudante a contribuir e fornecer um *feedback* baseado no seu julgamento da qualidade da solução apresentada pelo histórico do estudante que solicitou ajuda. Existe, também, um espaço denominado “Comentários”, na qual aparece o registro do diálogo entre os estudantes, possibilitando uma visualização adequada da conversa. Em seguida, o subsistema demonstra um espaço chamado “Novo comentário” que por meio de um editor facilita a colocação das contribuições pelos estudantes. Ao final da *interface*, existem dois botões, um para auxiliar no envio dos comentários e o outro para finalização do dialogo.

Figura 10 - O subsistema de *feedback* colaborativo com suas funcionalidades.




Fonte: Produção do próprio autor (2012).

O subsistema possui outras funcionalidades que auxiliam o estudante no processo de avaliação, como por exemplo, as informações inseridas no ambiente pelo estudante já com o diálogo aberto, são enviadas para o *email* dos estudantes que estão participando desse processo, facilitando a interação no ambiente e de um bloco de mensagens facilitando a organização das informações que será mencionado no item 3.3.1.1.

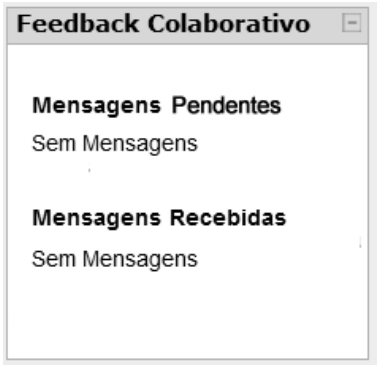
Posteriormente, as mensagens do subsistema de *feedback* podem ser socializadas pela turma, embora isso ainda não seja possível. Para o estudante que recebe o *feedback*, a mensagem pode reforçar, concorrer ou conflitar com a própria interpretação do estudante sobre os resultados alcançados.

3.3.1.1 Criação de um bloco de mensagens

O *Moodle* possui muitos recursos que facilitam o processo de ensino aprendizagem, um desses recursos são os blocos (*box*). Existem vários tipos de blocos nesse ambiente (administração, atividades, atividades recentes, calendário, mensagens e etc.) que atendem às demandas acadêmicas.

Com o desenvolvimento do subsistema de *feedback* colaborativo, surgiu a necessidade de criar uma forma prática para que os estudantes recebessem as informações que seriam produzidas pelo subsistema. A criação de um bloco para auxiliar as mensagens foi a estratégia adotada para o acompanhamento dessas mensagens pelo estudante. Ao criar o bloco, inicialmente, duas formas de controle foram adotadas, "Mensagem pendente" e "Mensagem recebida" como mostra a Figura 11.

Figura 11 - Bloco do Subsistema




Fonte: Produção do próprio autor (2012).

As mensagens socializadas entre os pares são disponibilizados nesse bloco, cada estudante possui sua identificação no ambiente, na qual o bloco criado reconhece as mensagens pendentes e recebidas desse estudante, facilitando dessa forma a organização das informações.

3.3.1.2 Planejamento e acompanhamento do professor

A opção de planejamento e mediação da aprendizagem foi uma opção adaptada as funcionalidades do *Moodle*. Além de planejar conteúdos e atividades, o professor utiliza o novo recurso para monitorar o diálogo pelos estudantes, sem intervenção, apenas auxiliando a evolução do processo de melhorando as condições de autoavaliar.

Especialmente com relação à avaliação automatizada, o professor pode visualizar instantaneamente os resultados (Figura 12), o que permite adotar diferentes estratégias para as dificuldades de aprendizagem apresentadas em seu mapa de acompanhamento, tais como criar duplas de estudantes durante as aulas práticas de laboratório.

Figura 12 - Visualização das respostas dos estudantes por questão pela visão do professor

| Pergunta | Texto da pergunta | Nome completo | Histórico das respostas | Melhor nota | Feedback Colaborativo |
|---|---|---|---|---|---|
| 1 | Escrever um programa que lê 1 valor b, calcula e mostra a área do quadrado de lado b. Fórmula: lado x lado.<br>Exemplo de como deve ser a saída no console:<br>area=valorcalculado, para isso, faça print("area="+valorcalculado); | | | | |
| | | [redacted] | #1(9.8), #2(10.0), #3(10.0), | 10.0 | |
| | | [redacted] | #1(0.0), #2(9.9), #3(9.9), #4(9.9), #5(9.9), | 9.9 | Ir para Conversa |
| | | [redacted] | | 0.0 | |
| | | [redacted] | #1(0.0), #2(9.9), | 9.9 | |
| | | [redacted] | #1(0.0), #2(0.0), #3(0.0), #4(0.0), #5(0.0), #6(9.9), #7(9.9), #8(9.9), #9(9.9), #10(9.9), | 9.9 | |
| | | [redacted] | #1(0.0), #2(0.0), #3(0.0), #4(0.0), #5(0.0), #6(0.0), #7(0.0), #8(10.0), #9(0.0), #10(10.0), #11(10.0), | 10.0 | Ir para Conversa |
| | | [redacted] | #1(9.9), | 9.9 | Ir para Conversa |
| | | [redacted] | #1(0.0), #2(9.9), | 9.9 | Ir para Conversa |
| | | [redacted] | #1(0.0), #2(9.9), #3(9.9), #4(10.0), | 10.0 | Ir para Conversa |
| | | [redacted] | #1(10.0), #2(9.9), #3(9.7), #4(9.9), #5(10.0), | 10.0 | |
| | | [redacted] | #1(9.9), | 9.9 | |
| | | [redacted] | | 0.0 | |
| | | [redacted] | | 0.0 | Ir para Conversa |
| | | [redacted] | | 0.0 | |

Fonte: Produção do próprio autor (2012).

A *interface* do ambiente de acompanhamento do professor desempenha as seguintes descrições: (1) o texto da pergunta; (2) o nome completo do estudante; (3) o histórico das respostas; (4) melhor nota e (5) o *feedback* colaborativo com a opção "ir para conversa", na qual o professor vai encontrar todo o diálogo entre os estudantes.

## 3.4 DESCRIÇÃO TÉCNICA DO SUBSISTEMA DE *FEEDBACK* COLABORATIVO

O Subsistema de *Feedback* Colaborativo foi desenvolvido em um ambiente totalmente em código aberto, ou seja, ferramentas que respeitam a liberdades definidas pela fundação que gerencia esses softwares, a FSF (Free Software Foundation). Para o desenvolvimento do subsistema foram utilizadas três ferramentas computacionais: o *WebService APACHE*, o bando de dados *MYSQL* e a linguagem *PHP*.

Segundo o sítio oficial da instituição (apache.org), o *APACHE* é um software de servidor HTTP (protocolo de transmissão de dados) responsável por disponibilizar conteúdo da web que pode ser acessado através da internet. No desenvolvimento do subsistema, ficou responsável pelo suporte do ambiente de desenvolvimento, no qual o *moodle* foi instalado para a implementação e modificações dos códigos fonte .

O sistema de gerenciamento de banco de dados foi o *MYSQL* que utiliza a linguagem SQL (Linguagem de Consulta Estruturada) através de pesquisa declarativa, o banco foi utilizado devido ser a base de desenvolvimento das ferramentas utilizados nessas pesquisas.

A linguagem utilizada foi o PHP, linguagem a qual foi desenvolvimento o *moodle*, sistema base desse trabalho.

# 4 ANÁLISE DE VIABILIDADE E USABILIDADE DO SUBSISTEMA

Foi ofertado um curso básico de programação para uma avaliação inicial de viabilidade do subsistema de *feedback* colaborativo no Ambiente de Aprendizagem de Programação. Segundo MAFRA et al (2006), esse tipo de estudo tem como objetivo principal permitir ao pesquisador avaliar se a aplicação da tecnologia é viável, ou seja, se atende de forma razoável aos objetivos inicialmente definidos, justificando a continuação do estudo.

Esta análise apresentou três momentos:

1. primeiro momento: oferta do curso de programação;
2. segundo momento: avaliação heurística;
3. terceiro momento: ensaio de interação;

Este capítulo relata a oferta do curso básico de programação, no qual os estudantes tiveram o primeiro contato com o subsistema, a avaliação de usabilidade e os resultados encontrado em cada método para alteração do sistema.

## 4.1 PRIMEIRO MOMENTO: OFERTA DO CURSO DE PROGRAMAÇÃO

O público-alvo do curso foi composto por estudantes na área de computação e pessoas interessadas em aprender programação. No curso, não houve uma seleção destes estudantes para sua matrícula. Em três dias após o início das inscrições, havia mais de 50 estudantes interessados. Para o estudo, 35 estudantes iniciaram o curso que possui de dez horas na modalidade a distância, em que os estudantes usavam o Ambiente de Programação na resolução das atividades propostas. O curso foi dividido em dois módulos, sendo que o primeiro tinha quatro horas e o segundo de seis horas.

A principal finalidade do curso básico de programação foi analisar o Subsistema de Feedback Colaborativo em estudantes que estivessem iniciando em programação por meio de um Ambiente de Ensino Aprendizagem.

Nesse curso, pretendeu-se avaliar a facilidade de uso do Subsistema de *feedback* verificando possíveis deficiências no ambiente e melhorando sua navegabilidade, este curso não teve a intenção de conceituar o conhecimento dos estudantes em programação.

O Curso Básico de Programação teve duração de uma semana e possuía dois módulos (Quadro 2). O módulo 1, possuía na ocasião 4h, em que foi apresentado o conteúdo programático

do curso através de instruções no tópico introdutória da sala virtual, com informações sobre o curso como experimento, as ferramentas que faziam parte do ambiente, exemplos com código prontos, exercícios com questões em *Java* e o questionários de programação. O módulo 2, com 6h, possuía as mesma instruções com um conteúdo  nível de dificuldade maior.

O conteúdo programático dos módulos do curso está apresentado a seguir.

Quadro 2 - Conteúdo programático do curso básico de programação

| Módulos | Conteúdo Programático |
|---|---|
| **Módulo 1** | • Introdução à programação;<br>• Algoritmos e programas;<br>• Constantes e variáveis, tipos de variáveis, declaração de variáveis;<br>• Operações e operadores: aritméticos, relacionais e lógicos;<br>• Entrada e saída de dados;<br>• Listas de exercícios do módulo 1;<br>• Questionário de Programação do módulo 1. |
| **Módulo 2** | • Estrutura de decisão *if*/*else*;<br>• Comandos *if's* concatenados;<br>• Estrutura de seleção *switch/case*;<br>• Estrutura de repetição *for*;<br>• Estrutura de repetição *while*;<br>• Estrutura de repetição *do*/*while*;<br>• Funções *Math*;<br>• Noções de *arrays* unidimensionais;<br>• Questionário de Programação do módulo 2; |

Fonte: Produção do próprio autor (2012).

No ambiente do curso foram disponibilizados dois tutores com conhecimento em programação *Java* para sanar as dúvidas dos estudantes, que foram auxiliados dentro dos recursos de fóruns e pelo *chat* (em tempo real). A Figura 12, mostra a sala virtual onde foi executado o curso básico de programação.

Figura 13 - Sala virtual do curso básico de programação

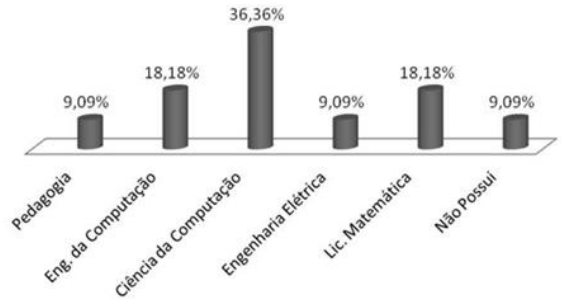

Fonte: Produção do próprio autor (2012).

O curso básico, também foi elaborado para testar a interatividade do estudante com o subsistema de *feedback*, sabendo que a interface é de fundamental importância durante esse processo de interação com o sistema, pois estabelece como e quais informações estarão disponíveis ao estudante. Para se obter uma relação amigável e facilitada no processo interativo, é necessária a construção de interfaces que obedeçam as conformidades ergonômicas dos perfis dos estudantes. Dessa forma, a usabilidade (USABILIDADE, 2011) é um aspecto a ser considerado, pois é por meio de uma boa usabilidade que estudantes e professores podem realizar o *feedback* de forma simples e eficaz.

### 4.1.1 Perfil dos estudantes

A divulgação do curso foi feita pela internet, utilizando ferramentas como lista de *emails* e rede social. Estudantes de várias áreas participaram do experimento, inicialmente foram inscritos 34 estudantes no *Moodle*, sendo que 12 estudantes responderam ao questionário inicial do ambiente. O Gráfico 1 exibe as informações sobre a graduação dos participantes do curso.

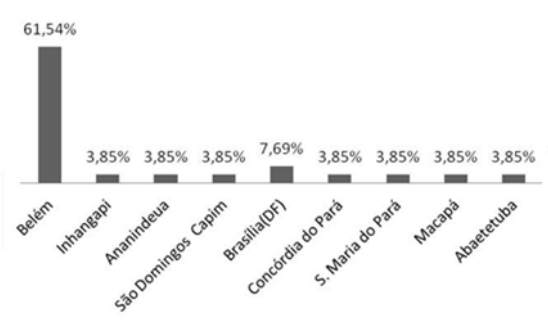
Gráfico 1 - Graduação dos participantes do curso de programação




Fonte: Produção do próprio autor (2012)

A divulgação do Curso Básico de Programação abrangeu principalmente estudantes de graduação. Além de estudantes da Universidade Federal do Pará (25), a divulgação começou, havia estudantes de vários institutos de ensino superior como Universidade Federal Rural da Amazônia (UFRA) (5) e Universidade de Brasília (UNB) (3) e de apenas uma faculdade particular a Estácio (FAP) (1). Além disso, a procedência dos participantes foi de diversas cidades, assim como o tipo de profissional e a idade, que teve com uma média de 28,5 anos. O Gráfico 2 mostrar a classificação dos participantes em relação à localidade.

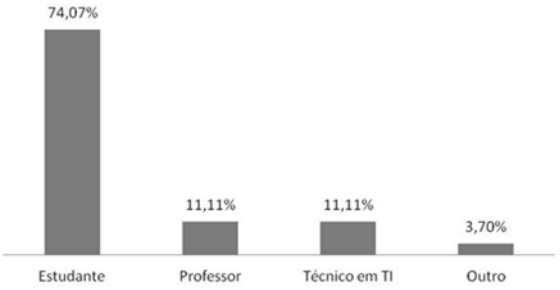
Gráfico 2 – Local residente do participante




Fonte: Produção do próprio autor (2012).

O gráfico 3, apresenta os resultados dos participantes em relação à profissão que cada um tinha durante a participação do curso.

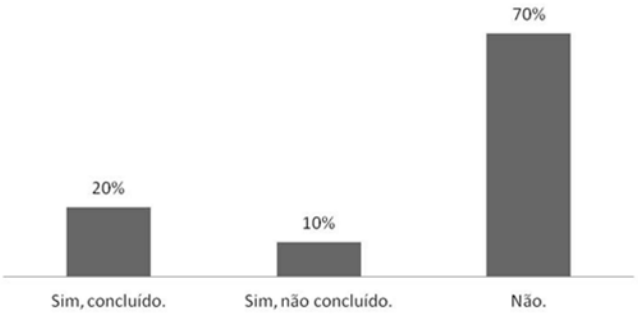
Gráfico 3 – profissão do participante do curso de programação




Fonte: Produção do próprio autor (2012).

Quanto à realização/conclusão de um curso técnico, a classificação dos estudantes é demonstrada no gráfico 4.

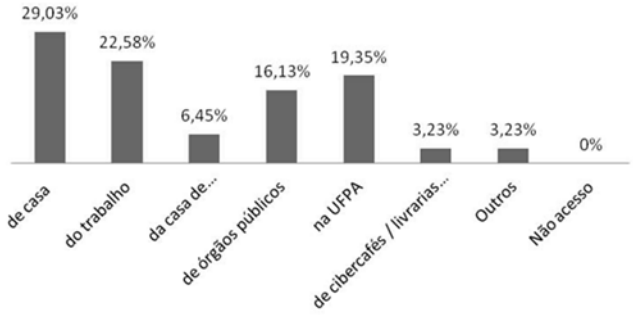
Gráfico 4 - Formação técnica dos participantes




Fonte: Produção do próprio autor (2012)

Sobre a utilização do computador, todos os estudantes declararam que possuem computador em casa, sendo que 88,89 % acessam à internet. O gráfico 5, representa os principais locais, no qual os participantes do curso acessam à rede mundial, considerando que o curso foi totalmente a distância, desta forma seria indispensável o acesso.

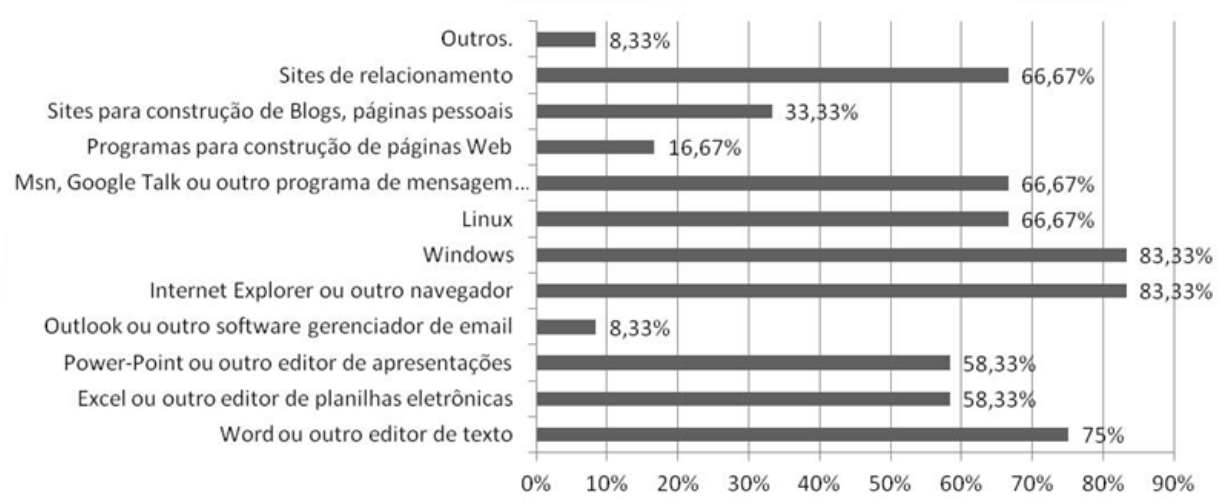
Gráfico 5 - Local de acesso a internet




Fonte: Produção do próprio autor

Em relação à utilização de programas, a maioria utiliza o navegador de *internet* principalmente *Internet Explorer* e o sistema operacional *Windows.* O Gráfico 6 mostra resultados da pergunta sobre que tipo de programas os participantes costumam utilizar.

Gráfico 6 - Programas mais utilizados pelos participantes




Fonte: Produção do próprio autor (2012).

Na análise de viabilidade, alguns resultados obtidos no estudo dos perfis dos participantes do curso nos apontam que 60% declararam terem sido incentivados no período do ensino médio a raciocinar de maneira lógica. Em levantamento, por volta de 41,67% qualifica como boa a capacidade de resolver problemas de raciocínios lógicos e que 66,67%, mais da metade dos estudantes afirmam que conseguem aprender estudando programas desenvolvidos por terceiros, incluindo códigos encontrados em livros, *Internet* e apostilas.

No questionário inicial, foi perguntado aos estudantes sobre o conhecimento de alguns dos conceitos fundamentais sobre programação, 98,96% dos estudantes disseram conhecer os conceitos

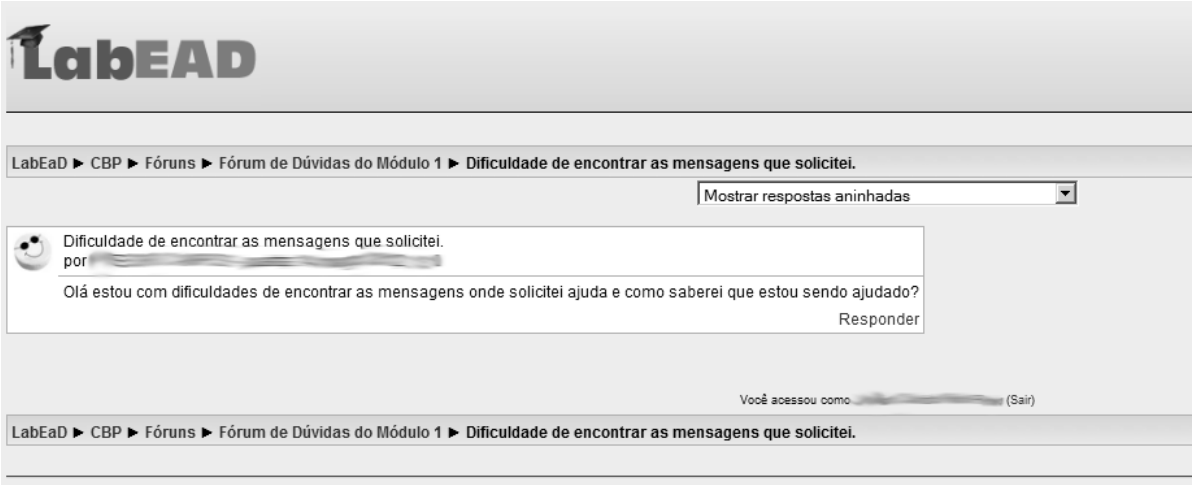
básicos de programação, saber o seu significado, mas não sabem como aplicar os conceitos da forma correta nos programas desenvolvidos.

Um dos itens relacionados ao questionário foi como preferem resolver problemas em pares (com a cooperação dos participantes da turma), mais da metade (52%) dos estudantes afirmam que gostam de desenvolver soluções em grupo. Enquanto isso, apenas 29% dos estudantes declaram ter facilidade em explicar suas soluções para os colegas. A utilização de ferramentas de auxílio ao ensino de programação não é amplamente conhecida entre os estudantes participantes do curso. Apenas 16% dos participantes dizem conhecer a existência dessas ferramentas.

### 4.1.2 Dificuldades relatadas pelos estudantes

Durante o Curso foram mencionadas uma série de dúvidas nos fóruns de ajuda, além de dificuldades em relação à interatividade com o subsistema de *feedback*, principalmente, com as mensagens de ajuda solicitadas pelo estudante e as mensagens de ajuda enviadas por outros estudantes. A Figura 14 ilustra um exemplo de dificuldade apresentada por um estudante.

Figura 14 - Mensagem de dúvida enviada ao fórum do módulo 1




Fonte: Produção do próprio autor (2012).

Outro problema relatado, foi referente à dificuldade de manuseio do subsistema de *feedback*. Alguns reclamaram das informações apresentadas no subsistema para enviar seus comentários, em que não entendiam que o botão de “enviar e encerrar diálogo”, servia apenas para finalizar a conversa, pois o subsistema não retornava para questionário principal, assim alguns diziam em fórum que não sabiam se o comentário com sua solução havia sido encaminhado para o seu destinatário.

A partir destas dificuldades e de outras (Anexo A), sentiu-se a necessidade de realizar uma análise de usabilidade no subsistema de *feedback* colaborativo dentro do ambiente virtual de ensino para facilitar aos estudantes a interação com o subsistema.

## 4.2 SEGUNDO MOMENTO: AVALIAÇÃO DA USABILIDADE

Este tópico, descreve como foi aplicado o método de usabilidade (Avaliação Heurística) para avaliação do subsistema de *feedback* colaborativo, de forma que os resultados dessas avaliações fossem usados para corrigir erros e melhorar o subsistema.

### 4.2.1 Usabilidade

Em virtude do crescimento contínuo da quantidade de informação na rede mundial de computadores, a exigência de produzir interfaces mais amigáveis devido ao grande número de usuários que hoje acessa à *internet*, colocou em evidência o termo usabilidade que se preocupa em estuda a facilidade de o usuário interagir com ferramentas ou objetos, a fim de realizar uma tarefa específica. A usabilidade tornou-se cada vez mais utilizada por projetistas de *web*. Segundo Bevan (1995), a usabilidade é o termo usado para descrever a qualidade da interação dos usuários com uma determinada interface.

De acordo com a *Internacional Organization for Standardization* (ISO/IEC), a usabilidade possui duas definições que caracterizam seus valores. A ISO 9126 (1991), define usabilidade como "Um conjunto de atributos de *software* relacionado ao esforço necessário para seu uso e para o julgamento individual de tal uso por determinado conjunto de usuários".

A ISO 9241 descreve que a usabilidade é compreendida enquanto "a capacidade de um produto ser usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso".

Segundo Nielsen (2003), a usabilidade está relacionada à cinco fatores de relação com o usuário, conhecido também como os cinco atributos da usabilidade:

- **Facilidade de aprendizado:** o usuário rapidamente consegue explorar o sistema e realizar suas tarefas.
- **Eficiência de uso:** tendo aprendido a interagir com o sistema, o usuário atinge níveis altos de produtividade na realização de suas tarefas.

- **Facilidade de memorização:** após um período sem utilizá-lo, o usuário não frequente é capaz de retornar ao sistema e realizar suas tarefas sem a necessidade de reaprender como interagir com ele.

- **Baixa taxa de erros:** o usuário realiza suas tarefas sem maiores transtornos e é capaz de recuperar erros, caso ocorram.

- **Satisfação subjetiva:** o usuário considera agradável à interação com o sistema e se sente subjetivamente satisfeito com ele.

A usabilidade possui regras que são ações executadas a partir das necessidades demonstradas pelo usuário, com a finalidade de melhorar o sistema. A estratégia de acompanhar os registros das áreas acessadas pelo usuário para determinar adaptações, compreende em uma maneira amplamente utilizada para conformidade com as necessidades e reações do usuário.

Para que essas ações possam ter efeito e as melhorias no sistema ocorram de forma satisfatória, resistências não devem ser encontradas pelo usuário, dificultando sua utilização em seu primeiro contato. Os procedimentos serão válidos a partir da confiança que estabelecer entre o usuário com o sistema.

#### 4.2.1.1 Tipos de avaliação de usabilidade

Para mencionar que um software está pronto para uso, é relevante saber se ele correspondido aos usuários, nas tarefas e no ambiente em que será aproveitado. Assim como testes de funcionalidade são indispensáveis para se verificar a robustez do sistema, a avaliação de interface é fundamental para se analisar a qualidade de uso de um *software*.

Métodos e princípios de um projeto de interface para um projetista não são garantias de uma alta qualidade de uso de seu *software*. Além disso, imaginar que os usuários pensem como o projetista, deve-se ter em mente que alguém avaliará a qualidade de uso do seu sistema.

Alguns dos principais objetivos de se realizar avaliação de sistemas interativos são:

- identificar as necessidades dos usuários ou verificar o entendimento dos projetistas sobre estas necessidades;

- identificar problemas de interação ou de interface;

- investigar como uma interface afeta a forma de trabalhar dos usuários;

- comparar alternativas de projeto de interface;

- alcançar objetivos quantificáveis em métricas de usabilidade;
- verificar conformidade com um padrão ou conjunto de heurísticas.

Segundo Rocha e Baranauskas (2003), de forma resumida, a avaliação têm três grandes objetivos:

- Avaliar a funcionalidade do sistema: é importante observar se o sistema está adequado aos requisitos da tarefa do usuário, ou seja, permite que o usuário execute sua tarefa de modo fácil e rápido.
- Avaliar o efeito da interface junto ao usuário: tentar identificar partes da interface que quando usadas causam resultados inesperados ou geram dúvidas nos usuários.
- Identificar problemas específicos do sistema: é necessário avaliar a usabilidade das interfaces, ou seja, avaliar a facilidade de uso, identificar áreas da interface que sobrecarregam o usuário de alguma forma, entre outros.

Estes objetivos classificam as técnicas de avaliação baseando-se em dois critérios: participação dos usuários nos testes, e se a interface está ou não implementada (ROCHA e BARANAUSKAS, 2003). De acordo com o critério, uma técnica deve ser escolhida para fazer a avaliação. Se os usuários não forem participar dos testes, a técnica mais indicada é a inspeção de usabilidade.

### 4.2.2 Resultados encontrados pela avaliação heurística utilizando a ferramenta HEVA

A inspeção de usabilidade possui métodos intimidadores, caros, difíceis e que necessitam de tempo para ser aplicado. Neste sentido, optou-se nesta pesquisa por um método barato, rápido e fácil de ser usado, tendo-se adotado a Avaliação Heurística para identificar os problemas de usabilidade na *interface*. A Avaliação Heurística foi realizada por três avaliadores especialistas ligados a área de computação. A escolha do método se deu principalmente pela sua facilidade de utilização e requisitar poucos avaliadores para analisar e corrigir problemas de usabilidade em interface.

Heurísticas são regras informais de julgamento que guiam pessoas numa rápida tomada de decisão. Deste modo, as heurísticas levam aos resultados esperados em menor tempo e por meio de melhores escolhas. A Avaliação Heurística consiste em realizar inspeção da interface tendo como base uma pequena lista de heurísticas de usabilidade.

Na fase de planejamento desse método, define-se de que forma a interface será exibida aos avaliadores: em que analisam as interfaces individualmente, verificando a semelhança da interface com as dez heurísticas reunidas por Nielsen (1990; 1994b):

- **Visibilidade do status do sistema:** o sistema deve sempre manter os usuários informados sobre o que está acontecendo com *feedback* apropriado e em um tempo razoável. A presença de *feedback* consegue também ajudar a reduzir a ansiedade do utilizador, permitindo que este saiba se a tarefa está ou não a ser realizada.

- **Compatibilidade entre sistema e mundo real:** o sistema deve utilizar a linguagem do usuário, com palavras, frases e conceitos familiares para ele, ao invés de termos específicos de sistemas. Seguir convenções do mundo real, fazendo com que a informação apareça em uma ordem lógica e natural.

- **Controle e liberdade para o usuário:** acredita-se que um sistema interativo dever permitir ao utilizador interagir quando e como deseja. No entanto, nem sempre se deve tentar cumprir ao máximo esta heurística.

- **Consistência e padrões:** referem-se ao fato de que os usuários não deveriam ter acesso a diferentes situações, palavras ou ações representando a mesma coisa. A interface deve ter convenções não-ambíguas.

- **Prevenção de erros:** os erros são as principais fontes de frustração, ineficiência e ineficácia durante a utilização do sistema.

- **Reconhecimento em lugar de lembrança:** tornar objetos, ações, opções visíveis e coerentes. O usuário não deve ter que lembrar informações de uma parte do diálogo para outra. Instruções para o uso do sistema devem estar visíveis ou facilmente acessíveis.

- **Flexibilidade e eficiência de uso:** a ineficiência nas tarefas pode reduzir a eficácia do usuário e causar-lhes frustração. O sistema deve ser adequado tanto para usuários inexperientes quanto para usuários experientes.

- **Projeto minimalista e estético:** os diálogos não devem conter informações irrelevantes ou raramente necessárias. Esta heurística torna-se importante pela necessidade de garantir um acesso simples e rápido à informação.

- **Auxiliar os usuários a reconhecer, diagnosticar e recuperar erros:** mensagens de erro devem ser expressas em linguagem natural (sem códigos), indicando precisamente o erro e sugerindo uma solução.

- **Ajuda e documentação:** mesmo que seja melhor que o sistema possa ser usado sem documentação, pode ser necessário fornecer ajuda e documentação. Tais informações devem ser fáceis de encontrar, ser centradas na tarefa do usuário, listar passos concretos a serem seguidos e não ser muito grandes. A ajuda deve estar facilmente acessível e on-line. O mapa do site se mostra uma boa ferramenta de ajuda, desde que ele cumpra o papel de orientar o usuário respondendo as perguntas do tipo: Onde estou? Onde estive? E onde posso ir?

Com base nas dez heurísticas compiladas por Nielsen, foi elaborado uma Avaliação Heurística, por três avaliadores externos, ligados a área, utilizando a ferramenta HEVA (OEIRAS et al, 2008). Esta ferramenta desenvolvida para dar suporte à realização de Avaliações Heurísticas de sistemas *Web*, tendo sido criada como uma extensão de um navegador, permitindo assim incluir novas funcionalidades, devido ao seu código aberto, esta por sua vez possibilita aos avaliadores capturar imagens da tela do computador, inserir comentários que fundamentam a criação de novas heurísticas específicas, já que tem a oportunidade de aprofundar a verificação, aplicando seus conhecimentos com propriedade afim de produzir soluções para os problemas apresentados.

A partir das avaliações, foram identificados vários problemas no subsistema de *feedback* colaborativo, os quais foram resumidos em quadros ilustrados (3 a 12) para melhor compreensão, o relatório completo encontra-se no anexo B.

Os quadros apresentam os seguintes itens:

- **Requisito violado:** dentre os vários requisitos apresentados neste trabalho, este item diz respeito ao problema encontrado como o requisito que não está presente na construção da página *web*.

- **Grau de Severidade:** representa o quão o problema interfere na usabilidade do sistema. Para poder distinguir a gravidade dos problemas encontrados é atribuído um valor, ou seja, quanto maior for esse valor, maior é a implicação no sistema. Os valores são em função do grau de severidade dos problemas de usabilidade:

    - Grau de severidade (1) Irrelevante: Não há necessidade imediata de solução.

- Grau de severidade (2) Simples: Problema de baixa prioridade (pode ser reparado)
- Grau de severidade (3) Médio: Problema de média prioridade (deve ser reparado)
- Grau de severidade (4) Grave: Problema de alta prioridade (tem que ser reparado)
- Grau de severidade (5) Catastrófico: Muito grave, tem que ser reparado com urgência.

- **Descrição do problema:** descreve a dificuldade encontrada na interação com do usuário com o computador.

O Quadro 3 refere-se ao problema de usabilidade da mensagem de solicitar ajuda.

Quadro 3 - Detalhamento dos problemas da mensagem de ajuda

| Requisito violado | Grau de Severidade | Descrição do problema |
|---|---|---|
| Estética e design minimalista Flexibilidade e eficiência de uso | (3) Problema de usabilidade menor | Ao clicar em Pedir/Ver Ajuda o usuário deveria ser automaticamente redirecionado para esta funcionalidade. Mensagens de confirmação só devem ser exibidas caso o usuário selecione uma opção que apresente um comportamento irreversível ou diferenciado. Exemplo: sair, enviar, cancelar.<br>A caixa exige mais um clique para realizar a operação, o que pode até mesmo evitar que o estudante utilize o recurso. |

Fonte: Avaliação Heurística, 2012

O Quadro 4, ilustra detalhes do problema encontrado em relação ao editor de comentários do subsistema.

Quadro 4 - Detalhamento dos problemas do comentário na ferramenta

| Requisito violado | Grau de Severidade | Descrição do problema |
|---|---|---|
| · Prevenção de erros<br>· Ajudar os usuários a reconhecer, diagnosticar e corrigir erros | (4) Problema de usabilidade grave | A frase "Escreva um comentário aqui..." não desaparece quando clica-se na caixa de edição de texto. Como este texto já está fixado o usuário pode simplesmente clicar em enviar sem digitar o seu comentário real, o que gerará o envio de um texto incorreto e que é salvo pelo sistema, como pode ser notado na área “Comentários” (comentário #1). Além disso, o sistema permite o envio de comentários vazios, em emitir nenhum tipo de mensagem de erro ou confirmação e também o salva (comentário #2). |

Fonte: Avaliação Heurística, 2012

O Quadro 5 descreve informações sobre os problemas encontrados na caixa lateral do bloco de mensagens do subsistema.

Quadro 5 - Detalhamento dos problemas na caixa de mensagens do subsistema

| Requisito violado | Grau de Severidade | Descrição do problema |
|---|---|---|
| · Compatibilidade do sistema com o mundo real<br>· Consistência e padrões<br>· Reconhecimento ao invés de relembrança | (3) Problema de usabilidade menor | Descrição do Problema: Os títulos das "seções" não estão claros. O título “Ajudas pendentes” dá a entender que são ajudas que o estudante solicitou que ainda não foram atendidas, mas trata-se de ajudas que ele precisa responder. Deveria haver três seções: uma para as ajudas solicitadas por outros estudantes que o estudante precisa responder, uma para as ajudas solicitadas pelo estudante que ainda não receberam respostas e outra para as ajudas solicitadas do estudante já com resposta. |

Fonte: Avaliação Heurística, 2012

O Quadro 6 relata os problemas encontrados principalmente no fluxo das informações do subsistema.

Quadro 6 - Detalhamento dos problemas na página do Subsistema

| Requisito violado | Grau de Severidade | Descrição do problema |
|---|---|---|
| · Reconhecimento ao invés de relembrança | (3) Problema de usabilidade menor | Descrição do Problema: Não há indicação de que estudante enviou qual comentário. |
| · Controle do usuário e liberdade<br>· Consistência e padrões | (3) Problema de usabilidade menor | Descrição do Problema: Não existe a possibilidade de um estudante apagar um comentário seu (caso tenha errado, por exemplo). |
| · Controle do usuário e liberdade | (3) Problema de usabilidade menor | Descrição do Problema: Quando se envia um comentário não existe nenhuma possibilidade na interface de fechar a tela de dúvidas e voltar ao questionário. |
| · Visibilidade do status do sistema<br>· Prevenção de erros | (4) Problema de usabilidade grave | Descrição do Problema: *Label* do botão dá a entender que apenas a tela será fechada, mas o sistema finaliza a dúvida toda. Sugestão: substituir *label* para algo parecido com Enviar Comentário e Finalizar Ajuda. |
| · Controle do usuário e liberdade<br>· Consistência e padrões | (4) Problema de usabilidade grave | Descrição do Problema: O sistema não oferece a opção Cancelar ou Voltar. Não permitindo que o usuário saia da tela de ajuda senão pelo botão do próprio navegador. Não há a possibilidade de voltar à página do questionário caso, por exemplo, o estudante tenha acessado o *link* da dúvida errada. Só há a possibilidade de voltar à página principal do curso. |

Fonte: Avaliação Heurística, 2012

O Quadro 7 as descrições dos problemas voltam-se mais para o conteúdo de ajuda e de algumas expressões de designer do subsistema.

Quadro 7 - Detalhamento dos problemas na caixa de novo comentário do subsistema

| Requisito violado | Grau de Severidade | Descrição do problema |
|---|---|---|
| · Help e documentação | (3) Problema de usabilidade menor | Descrição do Problema: O conteúdo da ajuda oferecida (os atalhos) não é tão importante para a caixa de texto "Novo comentário". A ajuda deveria descrever o funcionamento da caixa e aí sim poderia existir um link para os atalhos. |
| · Consistência e padrões | (1) Problema estética | Descrição do Problema: O conteúdo da ajuda oferecida (os atalhos) não é tão importante para a caixa de texto "Novo comentário". A ajuda deveria descrever o funcionamento da caixa e aí sim poderia existir um link para os atalhos. |
| · Consistência e padrões | (3) Problema de usabilidade menor | Descrição do Problema: O *label* do botão deveria estar apenas como "Enviar", já que se sabe que é um novo comentário |

Fonte: Avaliação Heurística, 2012

O Quadro 8 descreve o problema de conteúdo da informação do bloco de mensagem do subsistema.

Quadro 8 - Detalhamento dos problemas do bloco de mensagem

| Requisito violado | Grau de Severidade | Descrição do problema |
|---|---|---|
| · Compatibilidade do sistema com o mundo real<br>· Prevenção de erros | (5) Necessita correção imediata | Descrição do Problema: Supõe-se que o número entre parênteses deva indicar o número de ajudas pendentes e recebidas, no entanto este contador sempre está nulo. Isto pode fazer com que o usuário pense que não há nenhuma ajuda pendente, quando na verdade há. |

Fonte: Avaliação Heurística, 2012

O Quadro 9 a descrição do problema é sobre o ambiente de acesso como professor no subsistema.

Quadro 9 - Detalhamento dos problemas no ambiente de acesso do professor

| Requisito violado | Grau de Severidade | Descrição do problema |
|---|---|---|
| · Consistência e padrões<br>· Reconhecimento ao invés de relembrança | (3) Problema de usabilidade menor | Descrição do Problema: Ao acessar com perfil de professor os itens selecionados não trazem nenhuma informação porque o acesso às dúvidas é feito através dos questionários. Entretanto, a presença da caixa selecionada dá a entender que não há nenhuma dúvida. |

Fonte: Avaliação Heurística, 2012

O Quadro 10 traça um problema em relação ao bloco de mensagem, especificamente, no conteúdo de informação.

Quadro 10 - Detalhamento dos problemas no bloco de comentários do subsistema

| Requisito violado | Grau de Severidade | Descrição do problema |
|---|---|---|
| · Visibilidade do status do sistema | (5) Necessita correção imediata | Descrição do Problema: Não sei se isso é um simples problema de usabilidade ou se é funcional.<br>A questão é que ao clicar em "Ajudas Pendentes" o sistema redirecionou para uma tela vazia, ou seja, sem nenhum comentário. Uma vez que não aparece o problema a ser solucionado o estudante não tem como dar a solução. |

Fonte: Avaliação Heurística, 2012

O Quadro 11 expõe o problema de incompatibilidade do subsistema com alguns navegadores.

Quadro 11 - Detalhamento dos problemas no novo comentário em relação aos navegadores

| Requisito violado | Grau de Severidade | Descrição do problema |
|---|---|---|
| · Consistência e padrões | (4) Problema de usabilidade grave | Descrição do Problema: A caixa de texto se diferencia nos navegadores *Mozilla Firefox* 10 (Imagem 9A) e *Google Chrome* 17 (Imagem 9B). Considerando que são dois dos navegadores mais populares no mercado, recomenda-se que o sistema funcione igualmente nos dois navegadores. |

Fonte: Avaliação Heurística, 2012

O Quadro 12 continua a descrever problemas encontrados nos novos navegadores de *Web*.

Quadro 12 - Detalhamento dos problemas entre os navegadores de internet

| Requisito violado | Grau de Severidade | Descrição do problema |
|---|---|---|
| · Consistência e padrões<br>· Ajudar os usuários a reconhecer, diagnosticar e corrigir erros | (4) Problema de usabilidade grave | Descrição do Problema: Ao utilizar o *Google Chrome*, a cada vez que é acessada a seção de ajuda, o usuário se depara com a seguinte mensagem de erro. Recomenda-se a adaptação do sistema para este navegador (Imagem 10). A mensagem de erro está em inglês e não auxilia o usuário a entender o problema. Caso não seja possível adaptar o sistema, recomenda-se atualizar a mensagem de erro. |

Fonte: Avaliação Heurística, 2012

Após a validação dos dados contidos nos relatórios avaliativos, foram gerados demonstrativos que permitem a visualização das opiniões dos avaliadores e dos problemas encontrados a serem corrigidos.

O Quadro 13 demonstra as médias gerais originadas entre cada heurística da mesma área aplicada no relatório dos avaliadores, considerando apenas as impressões fornecidas pelos avaliadores, não se preocupando com os problemas a serem discutidos, embora estes tenham sido

extraídos a partir destes dados. Na avaliação heurística realizada foram encontrados 15 problemas, totalizando 26 violações das heurísticas.

Quadro 13 – Resumo em porcentagem dos requisitos violados no relatório final

| Heurísticas | Grau de severidade |
|---|---|
| **Heurística 1 - Visibilidade do status do sistema** | 7,69 % |
| **Heurística 2 - Compatibilidade do sistema com o mundo real** | 7,69 % |
| **Heurística 3 - Controle do usuário e liberdade** | 11,54 % |
| **Heurística 4 - Consistência e padrões** | 30,77 % |
| **Heurística 5 - Prevenção de erros** | 11,54 % |
| **Heurística 6 - Reconhecimento ao invés de relembrança** | 11,54 % |
| **Heurística 7 - Flexibilidade e eficiência de uso** | 3,85 % |
| **Heurística 8 - Estética e design minimalista** | 3,85 % |
| **Heurística 9 - Ajudar os usuários a reconhecer, diagnosticar e corrigir erros** | 7,69 % |
| **Heurística 10 - Help e documentação** | 3,85 % |

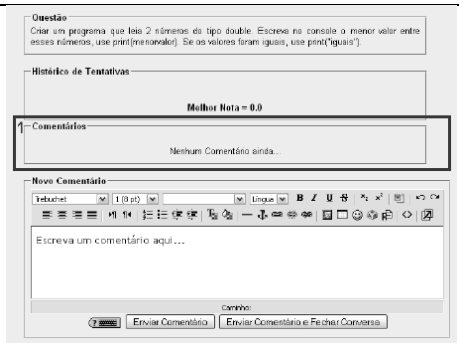
Fonte: Produção do próprio autor (2012).

Isto possibilitou a realização de uma análise categórica sobre a usabilidade implementada no subsistema de *feedback* colaborativo e, por conseguinte, determinou quais as melhorias necessitariam ser aplicadas ao subsistema.

### 4.2.3 Modificações realizadas no subsistema, a partir da Avaliação Heurística

As modificações realizadas no subsistema de *feedback* colaborativo foram baseadas no relatório final entregue pelos avaliadores após a avaliação heurística, tendo sido atribuídos níveis de prioridade aos problemas encontrados para correção do subsistema, no Quadro 14 mostra a descrição dos problemas encontrados pelo avaliador, com sua proposta de correção.

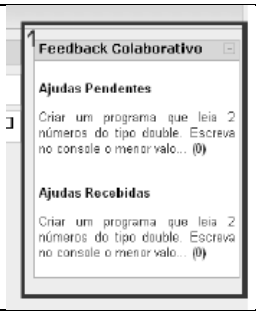
Quadro 14 – Grau de severidade 5 (necessidade de correção imediata)

| **Descrição do problema pelo avaliador**: | |
|---|---|
|  | (1) Não sei se isso é um simples problema de usabilidade ou se é funcional.<br>A questão é que ao clicar em "Ajudas Pendentes" o sistema redirecionou para uma tela vazia, ou seja, sem nenhum comentário. Uma vez que não aparece o problema a ser solucionado o estudante não tem como dar a solução. |
| **Resolução do problema:** O estudante só poderá enviar um comentário, preenchendo o campo necessário para envio do comentário, essa opção ficou obrigatória. | |

Fonte: Produção do próprio autor (2012).

O Quadro 15 demonstra a descrição dos problemas heurísticos encontrados pelos avaliadores no bloco de mensagem do subsistema de *feedback* colaborativo.

Quadro 15 - Grau de severidade 5 (necessidade de correção imediata) do bloco de mensagens

| **Descrição do problema pelo avaliador** | |
|---|---|
|  | (1) Supõe-se que o número entre parênteses deva indicar o número de ajudas pendentes e recebidas, no entanto este contador sempre está nulo. Isto pode fazer com que o usuário pense que não há nenhuma ajuda pendente, quando na verdade há. |
| **Resolução do problema:** Os contadores contabilizam a cada nova mensagem e agora as ajudas possuem a identificação do autor da mensagem. | |

Fonte: Produção do próprio autor (2012).

O Quadro 16 exibe problemas encontrados pelos avaliadores na avaliação heurística em relação aos comentários encontrados no subsistema, e a solução encontrada para resolver o problema.

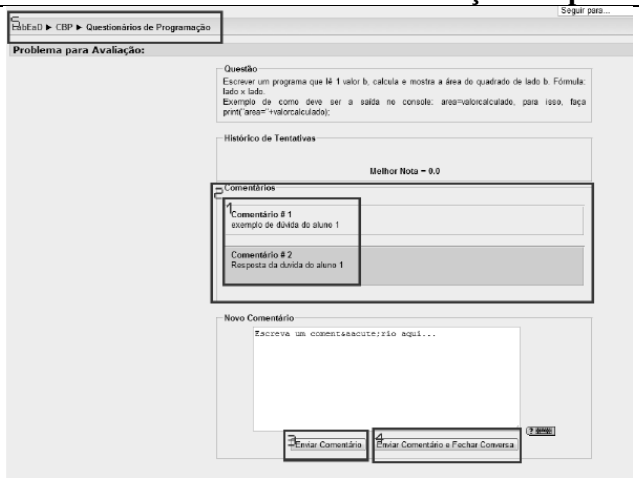
Quadro 16 - Grau de severidade 4 (problema de usabilidade grave) no subsistema

| **Descrição do problema pelo avaliador** | |
|---|---|
|  | A frase "Escreva um comentário aqui..." não desaparece quando clica-se na caixa de edição de texto. Como este texto já esta fixado o usuário pode simplesmente clicar em enviar sem digitar o seu comentário real, o que gerará o envio de um texto incorreto e que é salvo pelo sistema, como pode ser notado na área "Comentários" (comentário #1). Além disso, o sistema permite o envio de comentários vazios, em emitir nenhum tipo de mensagem de erro ou confirmação e também o salva (comentário #2). |
| **Resolução do problema:** No campo "Novo Comentário" a frase "Escreva um comentário aqui..." agora desaparece após colocar o cursor para escrever, agora é um campo obrigatório, ou seja, só poderá enviar comentário se o estudante digitar algo. | |

Fonte: Produção do próprio autor (2012)

O Quadro 17 apresenta as descrições dos problemas encontrados na página do subsistema, em seguida as soluções encontradas para solucionar cada um dos problemas.

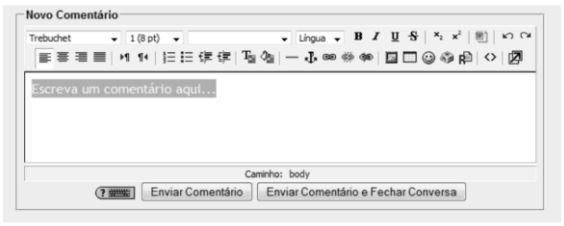
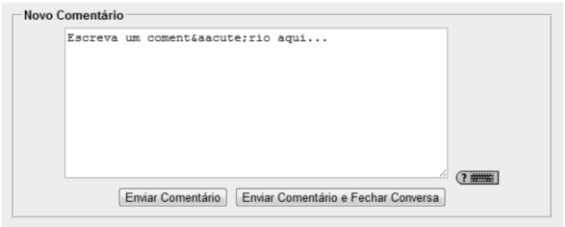
Quadro 17 - Grau de severidade 3 e 4 (problema de usabilidade menor e problema de usabilidade grave) encontrados no subsistema

| **Descrição dos problemas pelo avaliador:** | |
|---|---|
|  | (1) Não há indicação de que estudante enviou qual comentário.<br>(2) Não existe a possibilidade de um estudante apagar um comentário seu (caso tenha errado, por exemplo).<br>(3) Quando se envia um comentário não existe nenhuma possibilidade na interface de fechar a tela de dúvidas e voltar ao questionário.<br>(4) *Label* do botão dá a entender que apenas a tela será fechada, mas o sistema finaliza a dúvida toda. Sugestão: substituir *label* para algo parecido com Enviar Comentário e Finalizar Ajuda.<br>(5) O sistema não oferece a opção Cancelar ou Voltar. Não permitindo que o usuário saia da tela de ajuda senão pelo botão do próprio navegador. Não há a possibilidade de voltar à página do questionário caso, por exemplo, o estudante tenha acessado o *link* da dúvida errada. Só há a possibilidade de voltar à página principal do curso |
| **Resolução do problema:** (1) Os novos comentários possuem uma nova indicação de cor. (2) Adicionado uma opção de excluir o comentário. (3) Foi adicionado um botão para retornar ao questionário. (4) Adicionado no botão a opção de "Enviar Comentário e Finalizar Ajuda"; (5) Adicionado o botão de retornar ao questionário. | |

Fonte: Produção do próprio autor (2012).

O Quadro 18 mostra os problemas encontrados no subsistema de *feedback* colaborativo, especificamente, em relação à incompatibilidade com os navegadores de internet.

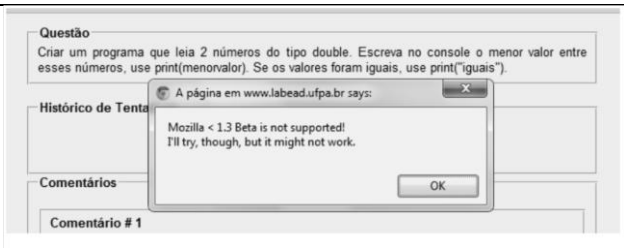
Quadro 18 – Grau de severidade 4 (problema encontrado com os navegadores de internet)

| **Descrição do problema pelo avaliador:** | |
|---|---|
| Imagem 9A - Caixa de texto no navegador Mozilla Firefox.<br>  <br><br>Imagem 9B - Caixa de texto no navegador Google Chrome<br>  <br> | A caixa de texto se diferencia nos navegadores *Mozilla Firefox* 10 (Imagem 9A) e *Google Chrome* 17 (Imagem 9B). Considerando que são dois dos navegadores mais populares no mercado, recomenda-se que o sistema funcione igualmente nos dois navegadores. |
| **Resolução do problema:** A opção de incompatibilidade foi corrigido por meio da atualização da plataforma moodle. | |

Fonte: Produção do próprio autor (2012).

O Quadro 19 mostra o problema no navegador *Google Chrome* e posterior solução.

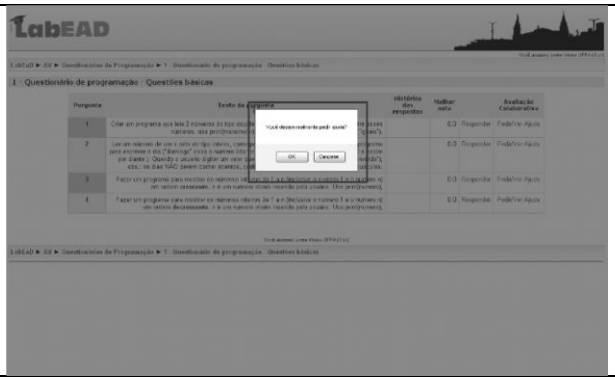
Quadro 19 – Grau de severidade 4 (problema encontrado com o navegador *Google Chrome*)

| **Descrição do problema pelo avaliador** | |
|---|---|
|  <br> | Ao utilizar o *Google Chrome*, a cada vez que é acessada a seção de ajuda, o usuário se depara com a seguinte mensagem de erro. Recomenda-se a adaptação do sistema para este navegador (Imagem 10).<br>A mensagem de erro está em inglês e não auxilia o usuário a entender o problema. Caso não seja possível adaptar o sistema, recomenda-se atualizar a mensagem de erro. |
| **Resolução do problema:** A opção foi corrigida após atualização do AVA moodle. | |

Fonte: Produção do próprio autor (2012).

O Quadro 20 revela o problema encontrado na funcionalidade dificultando o redirecionamento do subsistema para o estudante, com sua solução posterior.

Quadro 20 – Grau de severidade 3 (problema encontrado na funcionalidade do subsistema)

| **Descrição do problema pelo avaliador:** | |
|---|---|
|  | Ao clicar em Pedir/Ver Ajuda o usuário deveria ser automaticamente redirecionado para esta funcionalidade. Mensagens de confirmação só devem ser exibidas caso o usuário selecione uma opção que apresente um comportamento irreversível ou diferenciado. Exemplo: sair, enviar, cancelar.<br>A caixa exige mais um clique para realizar a operação, o que pode até mesmo evitar que o estudante utilize o recurso. |
| **Resolução do problema:** A opção foi retirada do subsistema. | |

Fonte: Produção do próprio autor (2012).

O Quadro 21 apresenta problemas encontrados pelos avaliadores nos títulos do bloco de mensagem do subsistema.

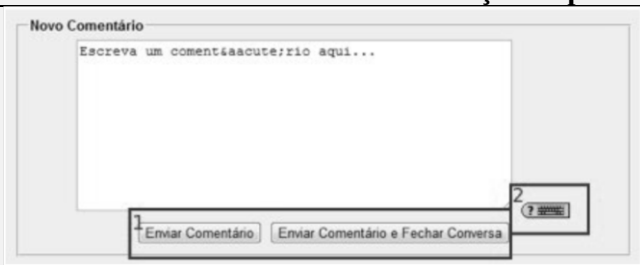
Quadro 21 – Grau de severidade 3 (problema nos títulos do bloco de mensagem)

| **Descrição do problema pelo avaliador** | |
|---|---|
|  | Os títulos das "seções" não estão claros. O título “Ajudas pendentes” dá a entender que são ajudas que o estudante solicitou que ainda não foram atendidas, mas trata-se de ajudas que ele precisa responder. Deveria haver três seções: uma para as ajudas solicitadas por outros estudantes que o estudante precisa responder, uma para as ajudas solicitadas pelo estudante que ainda não receberam respostas e outra para as ajudas solicitadas do estudante já com resposta. |
| **Resolução do problema:** Conforme o relatório, foram adicionados três seções no bloco do subsistema “Requisições de Ajuda”, no qual são as solicitações de ajuda por outros usuários. “Ajudas Recebidas” que são ajuda que você recebeu após usa ajuda e “Ajudas com Mensagens” são as conversas aberta que ainda não foram finalizadas. | |

Fonte: Produção do próprio autor (2012).

O Quadro 22 demonstra os problemas encontrados na avaliação heurística em relação ao título dos botões de envio e fechar do subsistema.

Quadro 22 – Grau de severidade 1,3 e 5 (problema nos títulos dos botões do subsistema)

| **Descrição do problema pelo avaliador:** | |
|---|---|
|  | (1) O conteúdo da ajuda oferecida (os atalhos) não é tão importante para a caixa de texto "Novo comentário". A ajuda deveria descrever o funcionamento da caixa e aí sim poderia existir um link para os atalhos.<br>**(2)** O *label* do botão deveria estar apenas como "Enviar", já que se sabe que é um novo comentário. |
| **Resolução do problema: (1)** O ambiente agora possui uma opção de *Help* explicando a funcionalidade do subsistema, está opção foi retirada. (2) foi adicionado um botão de "enviar". | |

Fonte: Produção do próprio autor (2012).

O Quadro 23 apresenta o problema encontrado no fluxo das informações no bloco de mensagem do subsistema de *feedback*.

Quadro 23 – Grau de severidade 3 (problema no fluxo das informações do bloco de mensagem)

| **Descrição do problema pelo avaliador** | |
|---|---|
|  | Ao acessar com perfil de professor os itens selecionados não trazem nenhuma informação porque o acesso às dúvidas é feito através dos questionários. Entretanto, a presença da caixa selecionada dá a entender que não há nenhuma dúvida |
| **Resolução do problema**: Foram retirados as opções do perfil do professor, o acompanhamento do professor será apenas pela opção dos questionários de avaliação. | |

Fonte: Produção do próprio autor (2012).

## 4.3 TERCEIRO MOMENTO: ENSAIO DE INTERAÇÃO

Esta pesquisa foi realizada em três partes. Primeiro foi ofertado um curso básico de programação, para observar a interatividade dos participantes com o subsistema. Segundo foi realizado uma avaliação heurística do subsistema de *feedback* colaborativo, seguindo os procedimentos e heurísticas propostos por Nielsen (1999) através de uma ferramenta denominada HEVA (OEIRAS et al., 2008). Após a avaliação foi gerado um relatório final dos três avaliadores sobre as heurísticas violadas, sendo que as mesmas foram posteriormente corrigidas no Subsistema.

Após as alterações realizadas no subsistema, seguindo os procedimentos da avaliação heurística, foi realizada uma nova avaliação do tipo ensaio de interação para verificar a ocorrência de mais problemas de usabilidade que não foram percebidos na Avaliação Heurística.

Os ensaios de interação são testes realizados com os usuários do sistema. É uma simulação de uso do sistema em que são apresentadas algumas tarefas para o usuário realizá-las. Os participantes que representam um público alvo são acompanhados pelos avaliadores que analisarão os comandos dados, os erros cometidos e o comportamento do usuário. Os ensaios de interação identificam problemas de interação de mais alto nível, dificilmente identificados por outros métodos.

Durante o processo de planejamento dessa avaliação, foi confeccionada uma lista de tarefas que visavam responder possíveis problemas no processo de interatividade. As tarefas a seguir compuseram o teste:

- Tarefa 1 : acessar o Ambiente Virtual de Ensino efetuando *login*.
- Tarefa 2 : acessar a sala de teste, copiar o código da atividade 1.
- Tarefa 3 : solicitar “Resposta” através do *Javatool* e colar a solução.
- Tarefa 4 : salvar a solução e solicite ajuda através do *Feedback* Colaborativo.
- Tarefa 5: no Subsistema de *Feedback* Colaborativo, descreva sua dúvida com relação a sua solução. Salve e aguarde seu solicitante enviar a colaboração.
- Tarefa 6 : após receber a solicitação de ajuda, agradeçer a colaboração e fechar a conversa.
- Tarefa 7 : efetuar o *logout*.

Para esse método, foram selecionados cinco usuários do curso básico de programação. Segundo Nielsen (1993), com este valor pode-se identificar aproximadamente 70% dos problemas mais críticos da *interface.*

Ao iniciar o teste, cada usuário recebeu um Termo de Consentimento (Apêndice B), leu e esclareceu suas dúvidas. O avaliador informou os objetivos do estudo, bem como o propósito do teste e os direitos do usuário, deixando-o livre para abandonar o teste e invalidá-lo a qualquer momento. Foi esclarecido que a privacidade do usuário seria preservada e que o objetivo de avaliação era melhorar a usabilidade do subsistema.

Após os participantes concordarem com o termo de consentimento e possuírem o manual de instrução, cada usuário recebeu a lista de tarefas e deu-se início o teste. A realização do ensaio obedeceu as seguintes ordens:

1) Antes da realização dos testes com os usuários, os dois avaliadores executaram os mesmos teste para ter a noção dos testes.

2) Cada usuário leu a lista de tarefas em voz alta.

3) Cada usuário teria que executar em cinco minutos a lista de tarefas.

Cabe ressaltar que o tempo total não deveria ultrapassar cinco minutos, embora em alguns momentos alguns usuários ultrapassarem o tempo e cada usuário era acompanhado por dois avaliadores enquanto realizavam as tarefas solicitadas. Alguns usuários desejaram fazer comentários a respeito de sua percepção sobre o Subsistema ao final da avaliação. Estes comentários também foram anotados.

A fotografia 1 mostra um participante realizando o ensaio de interação, com uma câmera gravando as expressões faceais.

Fotografia 1. Câmera focalizando as reações do rosto do usuário[2]

Fonte: Produção do próprio autor (2012).

A fotografia 2 , ilustra os avaliadores do ensaio de interação gravando a tela do computador do participante, captando as reações do participante em relação a tela do computador .

[2] A tarja foi inserida para preservar a identidade do participante no ensaio de interação.

Fotografia 2. Avaliadores observando as tarefas do usuário com a interface
Câmera focalizando a tela do computador

Fonte: Produção do próprio autor (2012)

Durante o teste algumas perguntas foram feitas para estimular o usuário a dizer o que estava pensado durante a sessão. Esta forma, conhecida como "*Thinking Aloud Protocol*", pensamento em voz alta, induz o usuário a expressar seus pensamentos de forma a captar suas opiniões. Abaixo estão relacionadas algumas das perguntas feitas aos participantes do ensaio:

- *"O que você pensou quando solicitou ajuda?"*
- *"Tem alguma coisa na interface do subsistema que você não gostou?"*
- *"O que você esperava da ajuda?"*

### 4.3.1 Resultados do ensaio de interação

De maneira geral, o subsistema alcançou às expectativas dos usuários, que relataram aprovar o ambiente de *feedback* colaborativo, sentiram-se confortáveis quanto à navegação e tiveram facilidade para encontrar as informações que procuravam.

O Quadro 24 mostra a porcentagem dos usuários que concluíram a lista de tarefas com sucesso, de forma que os objetivos foram atingidos em sua totalidade sem solicitar qualquer intervenção:

Quadro 24 – Porcentagem de tarefas completadas com sucesso

| Tarefa | Completadas com sucesso |
|---|---|
| **1** | 100% |
| **2** | 100% |
| **3** | 100% |
| **4** | 100% |
| **5** | 80% |
| **6** | 80% |
| **7** | 100% |

Fonte: Produção do próprio autor (2012)

Com relação ao Quadro 24, percebe-se que apenas duas tarefas não obtiveram o sucesso esperado, o que implica em 6% do total das tarefas, considerado um resultado positivo, visto as mudanças que já haviam realizados anteriormente no subsistema.

O Quadro 25 mostra os tempos médios para execução das tarefas pelos usuários e pelos avaliadores. O tempo do avaliador diz respeito ao melhor tempo encontrado entre os dois avaliadores, quando realizaram a lista de tarefas propostas aos participantes. O tempos esperado para resolução das tarefas foi apontado pelos avaliadores em função da complexidade destas. Já o tempo do usuário é através da media dos participantes do ensaio.

Quadro 25 – Porcentagem de tarefas completadas com sucesso (tempo)

| Tarefa | Tempo do Avaliador | Tempo esperado | Tempo do Usuário (Média) |
|---|---|---|---|
| **1** | 10s | 15s | 15s |
| **2** | 15s | 20s | 18s |
| **3** | 15s | 20s | 20s |
| **4** | 25s | 30s | 25s |
| **5** | 30s | 50s | 45s |
| **6** | 40s | 60s | 57s |
| **7** | 10s | 15s | 15s |

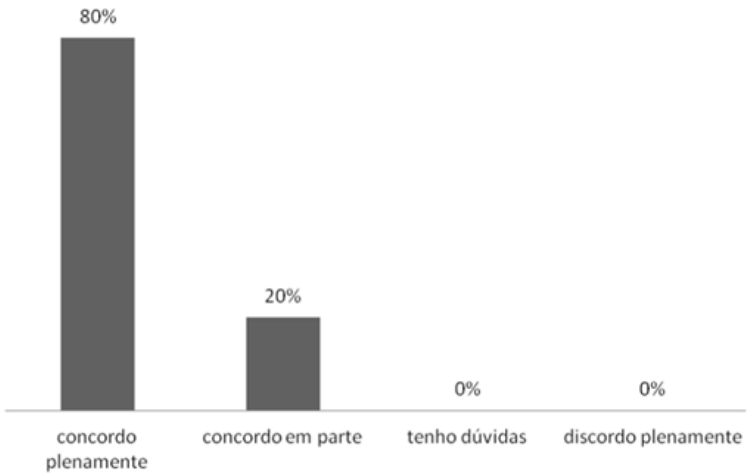
Fonte: Produção do próprio autor (2012).

Os dados do Quadro 25 indicam que os participantes concluíram as tarefas nos tempos previstos adendo para os resultados individuais. Infere-se que, as alterações ocorridas após a avaliação heurística foram de grandes contribuições para o desenvolvimento do subsistema.

Para a análise da satisfação, foi utilizado um questionário pós-teste entre os participantes do ensaio (Anexo B) cujo objetivo era coletar informações difíceis de identificar no momento da realização do ensaio de interação e possibilitar conhecer o sentimento do participante em relação ao subsistema de *feedback* colaborativo.

O Gráfico 7 exibe os resultados aprazíveis da interface do subsistema.

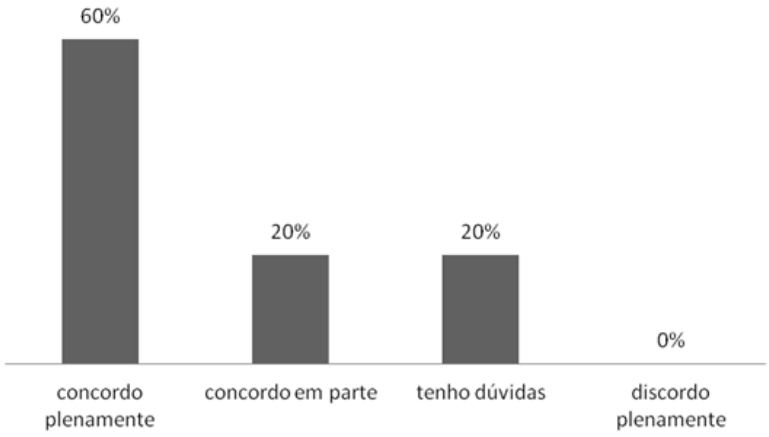
Gráfico 7 - a interface do software é agradável?




Fonte: Produção do próprio autor (2012).

O Gráfico 8 apresenta os resultados sobre a navegabilidade do subsistema de *feedback* com relação a sua facilidade de manuseio, desta forma, agradando 60% dos participantes.

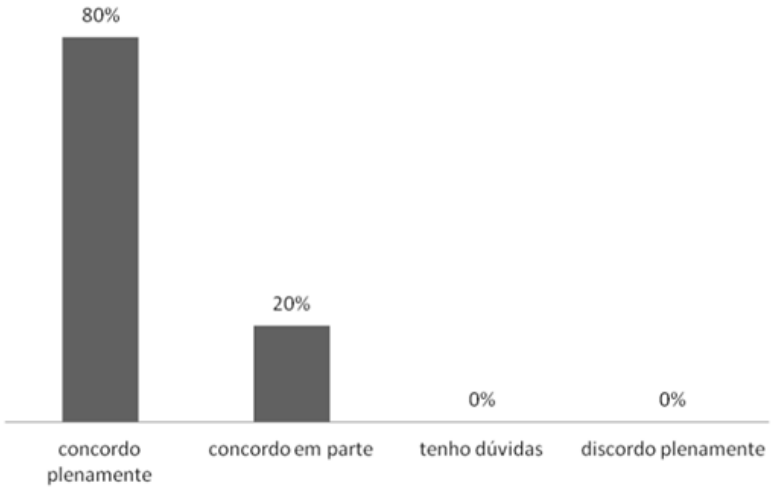
Gráfico 8 - Resposta à questão "a navegação no subsistema de feedback é fácil?"




Fonte: Produção do próprio autor (2012).

O Gráfico 9 demonstra os resultados sobre a clareza dos comandos demonstrados no subsistema de *feedback* colaborativo, sendo "concordado plenamente" para 80 % dos participantes do ensaio.

Gráfico 9 - Os comandos do subsistema são claros?




Fonte: Produção do próprio autor (2012).

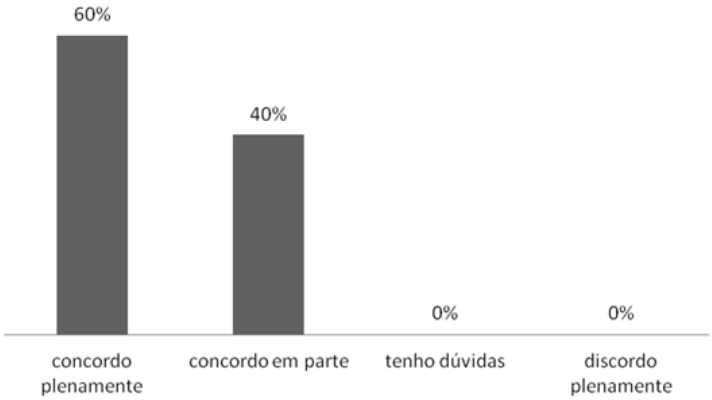
O Gráfico 10 apresenta os resultados com relação às dificuldades demonstradas pelos participantes do ensaio de interação para realização das tarefas solicitadas, 60% dos participantes afirmam não ter dificuldades em realizar as tarefas.

Gráfico 10 - não tive dificuldades em realizar tarefas


Fonte: Produção do próprio autor (2012).

O Gráfico 11 expões os resultados com relação à facilidade dos participantes em manusear o subsistema de *feedback* colaborativo, 60% afirmam que o subsistema é fácil de manusear e 40% concordam em parte.

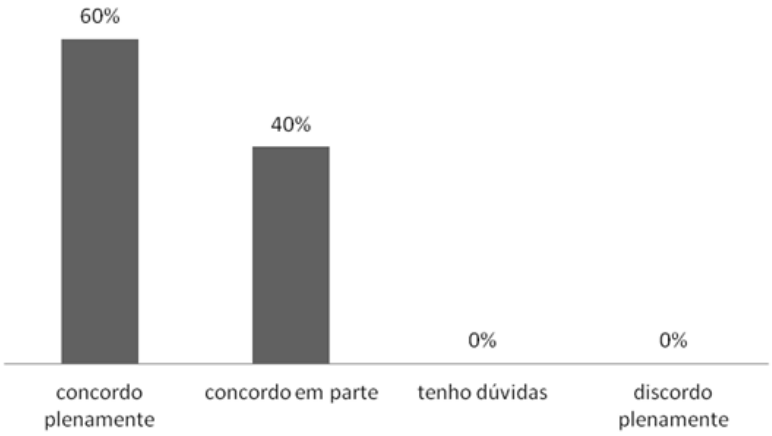
Gráfico 11 - Facilidade em usar o subsistema de feedback colaborativo.




Fonte: Produção do próprio autor (2012).

O Gráfico 12 demonstra os resultados sobre a satisfação dos participantes em relação ao subsistema de *feedback* colaborativo.

Gráfico 12 - Satisfação em usar Subsistema de *Feedback* Colaborativo




Fonte: Produção do próprio autor (2012).

A questão do instrumento de avaliação pós-teste solicitava ao estudante atribuir uma nota de 0 a 10 ao subsistema a média das notas atribuídas foi de 8,6 o que pode ser considerado um resultado positivo, embora ainda necessitando de melhoria.

### 4.3.2 Melhorias no subsistema de *feedback* colaborativo a partir dos resultados do ensaio de interação

Os resultados obtidos na Análise do Ensaio de Interação revelam que o subsistema realiza sua principal funcionalidade, facilitar as informações entre seus participantes no ambiente de programação. Mencionaram que o ambiente criado é agradável, sentiram-se confortável em relação

à navegabilidade devido o subsistema ser de fácil uso, obtiveram facilidade de encontrar as informações. Após o teste, poucos problemas foram encontrados, as melhorias são propostas dos próprios participantes buscando aperfeiçoar ainda mais o subsistema, vale ressaltar que essas melhorias são voltadas exclusivamente a interface do sistema.

Dentre as solicitações, está a melhoria das informações enviadas pelo *email* para os participantes do curso, em que o corpo da mensagem seja mais organizado para facilitar a compreensão. Outra sugestão foi em relação ao tamanho das fontes das mensagens localizadas no bloco de mensagem do subsistema, dois tiveram dificuldade de ler.

Em geral, as informações fornecidas pelos participantes acrescentaram ainda mais na melhoria do subsistema, souberam identificar os passos da lista de tarefas, sem nenhuma dificuldade.

# 5 CONSIDERAÇÕES FINAIS

Visando contribuir para a melhoria da formação de estudantes de cursos de computação, este trabalho apresentou importantes questões a serem abordadas e exploradas nas disciplinas de introdução a algoritmo e programação, à medida que cria mecanismos para valorizar a autoavaliação da aprendizagem. Para tal, a pesquisa teve como objetivo geral conceber e desenvolver um subsistema de *feedback* colaborativo para ser acoplado a um ambiente virtual de ensino de programação já existente e integrado à plataforma *Moodle*.

A ideia central é permitir aos estudantes explorar suas capacidades de autoavaliação de forma colaborativa, na perspectiva de que este novo ambiente auxilie numa melhor compreensão das abstrações dos códigos elaborados, ao mesmo tempo em que reduz o tempo do *feedback* entre eles.

Para o professor, acredita-se que a adoção deste novo ambiente oportunizará a adoção de outras abordagens didáticas, como por exemplo, as colaborativas, uma vez que ele possui acesso a um mapa de resultados, ainda que parciais, do desempenho do estudante durante a realização de seus estudos. Ao utilizar a avaliação automática, o professor visualiza os resultados, verificando o sucesso dos estudantes ou a necessidade de uma auxilio. A agilidade do *feedback* permite que o professor aprimore o planejamento de suas atividades ou aproveite de forma satisfatória a atuação dos monitores, de acordo com o mapa de evolução dos estudantes durante as práticas de laboratório.

Se anteriormente, apenas com o avaliador automático, era possível se obter ganhos na aprendizagem dos estudantes, à medida que (rapidez no *feedback*, reduz a sobrecarga do professor, diminui o tempo de espera pela avaliação de sua tarefa) com a inserção do subsistema de *feedback* colaborativo, além dos item citados, proporciona ao estudante a melhorar, refletir, questionar, compreender, planejar, expor ideias, ou seja, proporcionar suas próprias estratégias, buscam assim a melhor solução para seu problema.

Para a implementação deste subsistema, com foco na avaliação colaborativa, algumas etapas necessitaram ser alcançadas, dentre estas se destacam:

- integrar o subsistema ao ambiente de programação já existente;
- criar um espaço dentro do ambiente proposto para permitir o acompanhamento do professor do desempenho dos estudantes;
- implantar uma caixa lateral (tal qual as já existentes no *Moodle*) para acompanhamento das mensagens de *feedback;*

- criar mecanismos (avaliação da ferramenta por parte dos usuários, avaliação heurística e ensaio de interação) para analisar a viabilidade do subsistema de forma a prover subsídios para melhorias durante o próprio desenvolvimento deste;
- reconstruir e reestruturar a interface do subsistema, assim facilitando o ensino inicial em programação.

O esforço teórico empreendido durante a pesquisa, aliado aos resultados parciais obtidos das avaliações do subsistema, permitem conjecturar sobre suas contribuições não só para a área da computação, mas para outras áreas do conhecimento que necessitam da reflexão, da colaboração e do uso da lógica por parte dos estudantes. Assim, infere-se que trabalhar na perspectiva do *feedback* colaborativo proporcionará:

- melhorar o índice de aproveitamento dos estudantes iniciante nas disciplinas que possui a programação como base;
- proporciona a responsabilidade ao estudante na gestão da sua aprendizagem;
- possibilita a integração com outros participantes para dividir o trabalho;

## 5.1 DESAFIOS ENCONTRADOS E LIMITAÇÕES

Pelo fato de não terem sido encontrados na literatura referências a trabalhos similares e, também, devido ao subsistema proposto ter sido acoplado a um ambiente já existente, algumas demandas e dificuldades foram sentidas ao longo do processo de pesquisa. São estas:

- compreender as bases teóricas que dão sustentação ao trabalho;
- associar as teorias estudadas ao desenvolvimento do subsistema de feedback colaborativo;
- interpretar os resultados das pesquisas empíricas que fundamentaram a necessidade de elaboração da proposta do subsistema de feedback colaborativo;
- entender a funcionalidade do ambiente de programação existente, incluindo o *WebService* onde ele funcionava e, as ferramentas utilizadas (ex: Tomcat) para o seu desenvolvimento;
- compreender o simulador e visualizador de programas *Javatool*;
- apreender a funcionalidade do ambiente virtual *moodle*.

## 5.2 TRABALHOS FUTUROS

Como trabalhos futuros, melhorar continuamente o subsistema de *feedback*, sendo implementados novas melhorias e integrando com outras formas de avaliação e atividade dentro do ambiente *moodle* para que essa método posso melhorar não só a parte de programação de computadores e sim outras áreas do conhecimento.

Tratar níveis mais avançados do aprendizado de programação e incorporar o uso de outras linguagens, além de Java, como por exemplo:

- C.
- Ruby.
- Python.

Finalmente, como requisito não funcional, a meta é manter o ambiente portável para que novos módulos possam ser adicionados sem que haja problemas com as versões da plataforma *Moodle*.

## 5.3 ARTIGOS PUBLICADOS

Artigo publicado relacionado ao tema da dissertação.

SIROTHEAU, S. ; BRITO, S. R. de ; SILVA, Aleksandra Do S. ; Eliasquevici, Marianne K. ; FÁVERO, Eloi Luiz ; TAVARES, O. L. . Aprendizagem de iniciantes em algoritmos e programação: foco nas competências de autoavaliação. In: 22º. Simpósio Brasileiro de Informática na Educação, 2011, Aracaju/ES. Anais do 22º. Simpósio Brasileiro de Informática na Educação. Porto Alegre/RS : Sociedade Brasileira de Computação, 2011. p. 1-10.

## 5.4 ARTIGOS ACEITOS

Artigo aceito relacionado ao tema da dissertação.

Sirotheau, S. ; BRITO, S. R. de ; SILVA, Aleksandra Do S. ; Eliasquevici, Marianne K. ; FÁVERO, Eloi Luiz ; TAVARES, O. L. . Beginners in learning algorithms and programming: focus on self-assessment skills. In: International Conference on Web Information Systems and Computing Education (ICWISCE 2011), 2011, Bangkok, Thailand. Proceedings of ICWISCE, 2011. v. 1. p. 1-6.

## 6 REFERÊNCIAS


ALMEIDA, E. et al. AMBAP: um ambiente de apoio ao aprendizado de programação. In: WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO, 10. Florianópolis: SBC. 2002.

ANDERSON, J. The Rules of Thought. In: LEVY, B; SCHREIBER, E. S. **Secrets of the Mind**. CD-Rom Montparnasse Multimedia, Ubi Soft Hypermind, 2000.

BANDURA, A. **Social foundantions of thought e action:** a social cognitive theory. Englewood Cliffs: Prentice Hall, 1986.

BARR, R. B.; TAGG, J. A new paradigm for undergraduate education. **Learning from Change**, v. 27, n. 6, p. 13-25, 1995.

BEVAN, N. Human-computer interaction standards. In: ANZI; OGAWA. INTERNATIONAL CONFERENCE ON HUMAN COMPUTER INTERATION, 1995, Yokohama.

BLÜMKE, R. A. **A Experimentação no ensino de física**. Universidade Regional do Noroeste do Estado do Rio Grande do Sul – Departamento de Física, Estatística e Matemática. Ijuí, 2002.

BORGES, M. A. F. **Avaliação de uma metodologia alternativa para a aprendizagem de programação**. In: WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO. Curitiba, PR: SBC, 2000.

BOUD, D. Sustainable assessment: rethinking assessment for the learning society. **Studies in Continuing Education**, v. 22, n. 2, p. 151-167, 2000.

BRITO, S. R. et al. Computer Supported Collaborative Learning for helping novice students acquire self-regulated problem-solving skills in computer programming. In: THE 2011 INTERNATIONAL CONFERENCE ON FRONTIERS IN EDUCATION: Computer Science and Computer Engineering (FECS'11), 2011, Las Vegas: Worldcomp, 2011. v. 7.

BROWN, M. H. **Algorithm Animation**. Cambridge MA: MIT Press, 1988.

BUTLER, D. L.; WINNE, P. H. Feedback and self-regulated learning: a theoretical synthesis. **Review of Educational Research**, v. 65, n. 3, p. 245-281, 1995.

CASTRO, T. H. C.; CASTRO JR., A. N.; MENEZES, C. S. Aprende: um Ambiente Cooperativo de Apoio à Aprendizagem de Programação. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 15., 2004, Manaus – AM, **Anais**... Manaus: EDUA, 2004. v. 1, p. 71 – 79.

DWECK, C. **Self-theories**: their role in motivation, personality and development. Philadelphia: Psychology Press, 1999.

GIRAFFA, L.; MARCZAK, S.; ALMEIDA, G. O ensino de algoritmos e programação mediado por um ambiente na Web. In: CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 2003, Campinas, SP: SBC. 2003.

GALIAZZI, M. C. et al. Objetivos das atividades experimentais no ensino médio: a pesquisa coletiva como modo de formação de professores de ciências. **Ciência e Educação**, v. 7, n. 2, p. 249-263, 2001.

HIGGINS, R.; HARTLEY, P.; SKELTON, A. Getting the message across: the problem of communicating assessment feedback. **Teaching in Higher Education**, v. 6, n. 2, p. 269-274, 2001.

HUNDHAUSEN, C. D.; DOUGLAS, S. A.; STASKO, J. T. A meta-study of algorithm visualization effectiveness. **Journal of Visual Languages & Computing**, v. 13, n. 3, p. 259-290, 2002.

IVANIC, R.; CLARK, R.; RIMMERSHAW, R. What am I supposed to make of this? The messages conveyed to students by tutors' written comments**.** In: LEA, M. R.; STIERER, B. (Eds). **Student writing in higher education**: new contexts. Buckingham, SHRE: Open University Press, 2000.

KNOX, D. et al. **Use of laboratories in Computer Science education:** guidelines for good practice. USA: ACM – Association for Computing Machinery, 1996.

KULYK, O.; KOSARA, R.; URQUIZA-FUENTES, J.; WASSNICK, I. Human-centered visualization environments. **Lecture notes in computer science.** Chapter Human-Centered Aspects: Springer-Verlag, v. 4417, p. 13–75, 2007.

LAZAKIDOU, G.; RETALIS, S. Using computer supported collaborative learning strategies for helping students acquire selfregulated problem solving skills in Mathematics. **Computers & Education**, v. 54, n. 1, p. 3-13, 2010.

LEA, S. J.; STEPHENSON, D.; TROY, J. Higher education students' attitudes to student-centred learning: beyond 'educational bulimia. **Studies in higher education**, v. 28, n. 3, p. 321-334, 2003.

LEA, M.; STREET, B. V. Student writing and staff feedback in higher education: an academic literacies approach. **Studies in Higher Education**, v. 23, n. 2, p. 157-72, 1998.

MAFRA, S. N.; BARCELOS, R. F.; TRAVASSOS, G. H. Aplicando uma metodologia baseada em evidência na definição de novas tecnologias de software. In: Simpósio Brasileiro de Engenharia de Software, 20., 2006, Florianópolis, SC. **Anais...** 2006.

MENEZES, C. S.; TAVARES, O.L.; NEVADO, R.A.; CURY, D. Computer Supported Co-operative Systems to support the problem solving: a case study of learning computer programming. In: THE 38TH ANNUAL FRONTIERS IN EDUCATION (FIE) CONFERENCE, 2008, New York. v. 1.

MOODLE 2010: course management system. Disponível em: <http://moodle.org/>. Acesso em: 22 jun. 2011.

MOREIRA M. P.; FAVERO, E. L. Um ambiente para ensino de programação com feedback automático de exercícios. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO - CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 18., 2008, Belém-PA: SBC, **Anais...** Belém, 2009.

MORENO, A.; MYLLER, N.; SUTINEN, E.; BEN-ARI, M. **Visualizing Programs with Jeliot 3.** In Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI 2004) (pp. 373-376). Gallipoli, Lecce, Italy.

MOTA, M. P.; BRITO, S. R.; MOREIRA, M. P.; FAVERO, E. L. Ambiente integrado à plataforma moodle para apoio ao desenvolvimento das habilidades iniciais de programação. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 20., 2009, Florianópolis, **Anais...**, Florianópolis, 2009.

MOTA, M. P.; PEREIRA, L. W. K.; FAVERO, E. L. Javatool: uma ferramenta para ensino de programação. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO - CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 18. 2008, **Anais...** Porto Alegre-RS: Sociedade Brasileira de Computação, 2008.

NAPS, T. L. et al. Exploring the role of visualization and engagement in computer science education. **SIGCSE Bull**. v. 35, n. 2, p. 131-152, Jun. 2003.

NICOL, D. Principles of good assessment and feedback: theory and practice. In: From the REAP International Online Conference on Assessment Design for Learner Responsibility, 29th-31st May, 2007. Disponível em: <http://ewds.strath.ac.uk/REAP07>. Acesso em: 14 ago. 2012.

NICOL, D. J.; MACFARLANE-DICK, D. Formative assessment and self-regulated learning: a model and seven principles of good feedback practice. **Published in Studies in Higher Education**, v. 31, n. 2, p. 199-218, 2006.

NIELSEN, J. **Changes in web usability since 1994**. Dec. 1997. Disponível em: <http//www.useit.com/alertbox/9712a.html>. Acesso em: 18 mar. 2012.

_____. Desingning web usability. Indianápolis: New Riders, 1999.

_____. Projetando web sites. Rio de Janeiro: Campus, 2000.

_____. Usability Engineering. San Diego: Academic Press, 1993.

OEIRAS, J. Y.; BENTOLILA, D. L.; FIGUEIREDO, M. C. Heva: uma ferramenta de suporte à avaliação heurística para sistemas web. In: PROCEEDINGS OF THE BRAZILIAN SYMPOSIUM ON HUMAN FACTORS IN COMPUTING SYSTEMS, 8., 2008, Porto Alegre, RS, **Anais…**, Porto Alegre, RS, 2008.

PERRENOUD, P. **Avaliação da excelência à regulação das aprendizagens**: entre duas lógicas. Porto Alegre: Artmed, 1999.

PINTRICH, P. R. Understanding self-regulated learning. **Journal of new directions for teaching and learning**, v. 63, p. 3-12, 1995.

POLYA, G., **The Art of Solving Problems**. Rio de Janeiro: Interciência, 1978.

PROULX, V. K. Programming patterns and design patterns in the introductory computer science course. In: **Proceedings of the 31st SIGCSE.** Auxtin, USA, 2000. p. 7-12.

ROCHA, H. V.; BARANAUSKAS, M. C. C. **Design e avaliação de interfaces humano-computador**. Campinas, SP: NIED/UNICAMP, 2003.

RUST, C.; PRICE, M.; O'DONOVAN, B. Improving students learning by developing their understanding of assessment criteria and processes. In: **Assessment and Evaluation in Higher Education**, v. 28, n. 2, p. 147-164, 2003.

SADLER, D. R. Formative assessment and the design of instructional systems. **Instructional Science**, v. 18, p. 119-144, 1989.

SADLER, D. R. Formative assessment: revisiting the territory. **Assessment in Education**, v. 5, n. 1, p. 77-84, 1998.

SANTOS, L. Auto-avaliação regulada: porquê, o quê e como? In: ABRANTES, Paulo; ARAÚJO, Filomena (Org.). **Avaliação das aprendizagens**: das concepções às práticas. Lisboa: Ministério da educação, Departamento do Ensino Básico, 2002. p. 75-84. Disponível em: http://www.educ.fc.ul.pt/docentes/msantos/textos/DEBfinal.pdf. Acesso em: 22 jan. 2012.

SCHUNK, D. H. Self-efficacy and cognitive skill learning. In: AMES, Carol; AMES, Russell (Eds.). **Research on motivation in education**: goals and cognititons. New York: Academic Press, 1989. v. 3, p. 13-44.

SCHUNK, D. H.; ZIMMERMAN, B. J. Self regulation in education: retrospect and propect. In: SCHUNK, D. H.; ZIMMERMAN, B. J. **Self-regutation of learning and performance**: issues and educational applications. Hillsdale, NJ: Erlbaum, 1994.

SHANG, Y.; SHI, H.; CHEN, S. An intelligent distributed environment for active learning. **ACM Journal of Educational Resources in Computing** (JERIC). New York: ACM Press, v. 1, n. 2, Article 4, 2001.

STAMOULI, I.; HUGGARD, M. Object oriented programming and program correctness: the students perspective. In: INTERNATIONAL COMPUTING EDUCATION RESEARCH WORKSHOP, 2. United Kingdo: ACM, 2006.

TOBAR, C. M. et al. Uma arquitetura de ambiente colaborativo para o aprendizado de programação. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 12. Vitória, ES. **Anais**... Vitória, 2001.

**Usabilidade na website**. Disponível em:.<http://www.seisdeagosto.com/definicoes/usabilidade/usabilidade.php>. Acesso em: 3 abr. 2011.

YORKE, M. Formative assessment in higher education: moves towards theory and the enhancement of pedagogic practice. **Higher Education**, v. 45, n. 4, p. 477-501, 2003.

ZIMMERMAN, B. J. Self-efficacy: an essential motive to learn. **Contemporary Educational Psychology**, v. 25, n. 1, p. 82-91, 2000.

ZIMMERMAN, B. J.; SCHUNK, D. H. **Self-regulated learning and academic achievement: theoretical perspectives.**, New Jersey: Lawrence Erlbaum Associates, 2001.

**ANEXO A -** Dificuldades relatadas pelos estudantes

1. Os estudantes relataram dificuldades em relação a falta de um manual de ajuda com o passo a passo do subsistema de feedback colaborativo.
    a. Os estudantes relataram uma grande dificuldade com a falta de um material de apoio em relação ao Subsistema de *Feedback* Colaborativo, alguns colocaram nos fóruns que precisaria de ajuda para entender o sistema, pois não tinha nenhum material explicando o funcionamento do sistema.
2. Dificuldade com algumas notas que o sistema atribuía, relataram ficar inconformados com alguns resultados e surpreendidos com outras (por exemplo, o subsistema atribuiu notas acima da nota máxima).
3. Dificuldades com alguns navegadores, por motivo de versões do Applet do Java.
4. Problemas com algumas resoluções submetidas e o resultado do exibido pelo subsistema de feedback colaborativo..

**ANEXO B -** Avaliação do Sistema de *Feedback* Colaborativo

# RELATÓRIO DE AVALIAÇÃO HEURÍSTICA

**FEVEREIRO 2012**

**BELÉM -PA**

**1. Descrição do software avaliado**

Resposta - Sistema de *feedback* colaborativo para cursos do *Moodle*. O sistema permite que estudantes solucionem dúvidas entre si,com possíveis intervenções do professor.

**2. Público-Alvo do software**

Resposta - Professores e estudantes de cursos do *Moodle.*

**3. Quantidade de sessões de avaliação**

Resposta - 3

**4. Funcionalidades avaliadas**

Resposta - Envio de dúvida, resposta à dúvida e intervenção do professor.

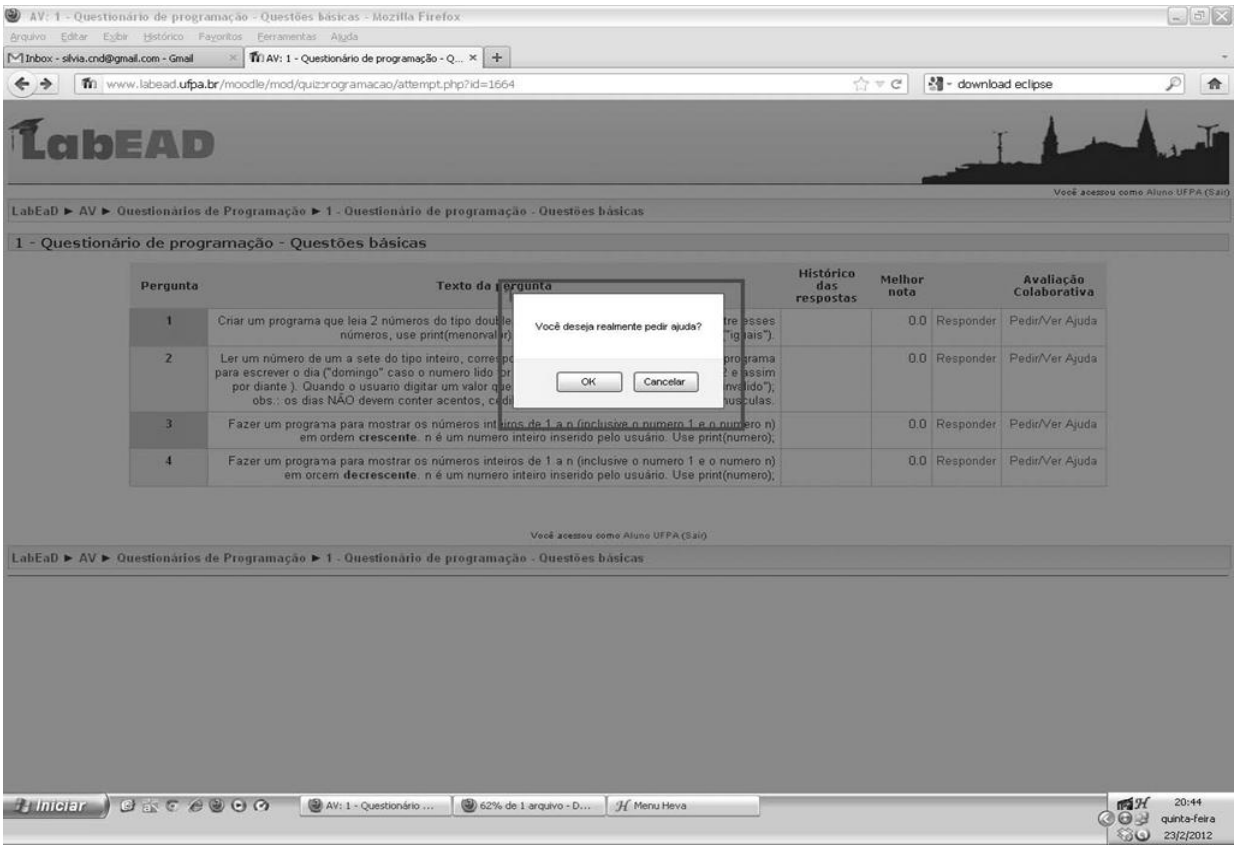
**5. Heurísticas violadas**

Resposta - Na avaliação heurística realizada foram encontrados 15 problemas totalizando 26 violação(ões) das heurísticas segundo o esquema abaixo:

Heurística 1 -Visibilidade do status do sistema: 2 violação(ões)
Heurística 2 -Compatibilidade do sistema com o mundo real: 2 violação(ões)
Heurística 3 -Controle do usuário e liberdade: 3 violação(ões)
Heurística 4 -Consistência e padrões: 8 violação(ões)
Heurística 5 -Prevenção de erros: 3 violação(ões)
Heurística 6 -Reconhecimento ao invés de relembrança: 3 violação(ões)
Heurística 7 -Flexibilidade e eficiência de uso: 1 violação(ões)
Heurística 8 -Estética e design minimalista: 1 violação(ões)
Heurística 9 -Ajudar os usuários a reconhecer, diagnosticar e corrigir erros: 2 violação(ões)
Heurística 10 -Help e documentação: 1 violação(ões)

## 5.1 Detalhamento dos problemas encontrados

Imagem 1



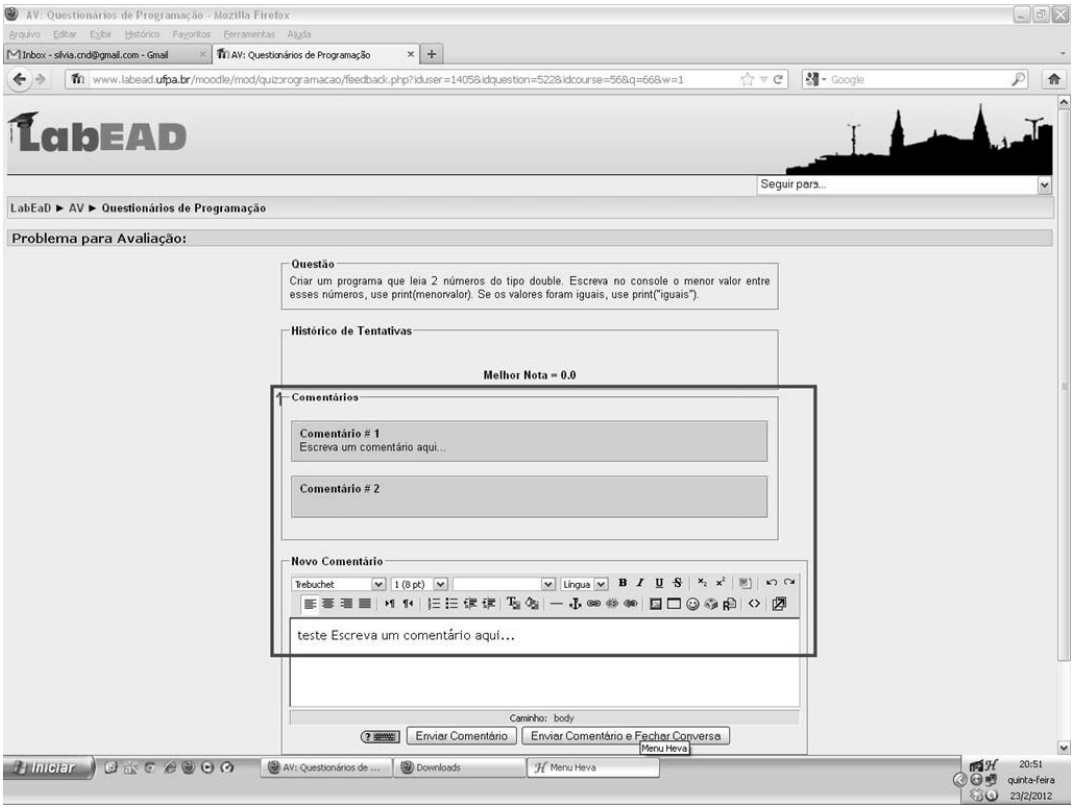

**Seleção 1**

**Heurística:** Estética e design minimalista

**Heurística:** Flexibilidade e eficiência de uso

**Grau de Severidade:** 3 -Problema de usabilidade menor

**Descrição do Problema:** Ao clicar em Pedir/Ver Ajuda o usuário deveria ser automaticamente redirecionado para esta funcionalidade. Mensagens de confirmação só devem ser exibidas caso o usuário selecione uma opção que apresente um comportamento irreversível ou diferenciado. Exemplo: sair, enviar, cancelar. A caixa exige mais um clique para realizar a operação, o que pode até mesmo evitar que o estudante utilize o recurso.



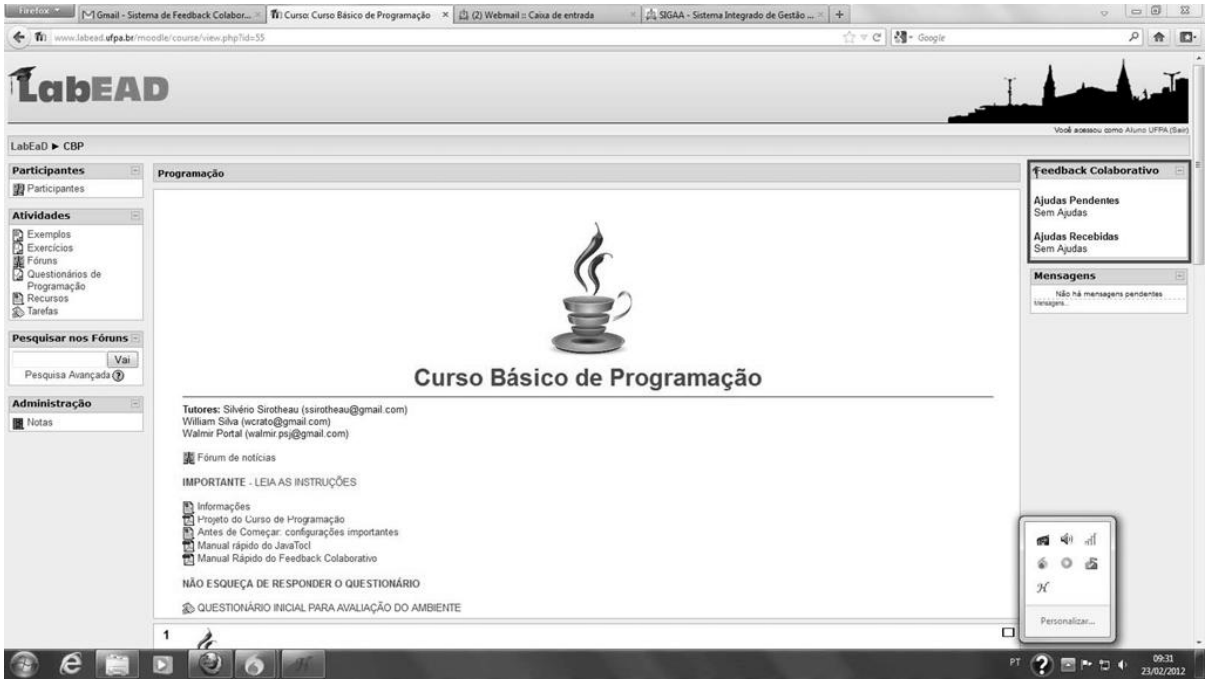

**Seleção 1**

**Heurística:** Prevenção de erros

**Heurística:** Ajudar os usuários a reconhecer, diagnosticar e corrigir erros

**Grau de Severidade:** 4 -Problema de usabilidade grave

**Descrição do Problema:** A frase "Escreva um comentário aqui..." não desaparece quando clica-se na caixa de edição de texto. Como este texto já esta fixado o usuário pode simplesmente clicar em enviar sem digitar o seu comentário real, o que gerará o envio de um texto incorreto e que é salvo pelo sistema, como pode ser notado na área “Comentários” (comentário #1). Além disso, o sistema permite o envio de comentários vazios, em emitir nenhum tipo de mensagem de erro ou confirmação e também o salva (comentário #2).




**Seleção 1**

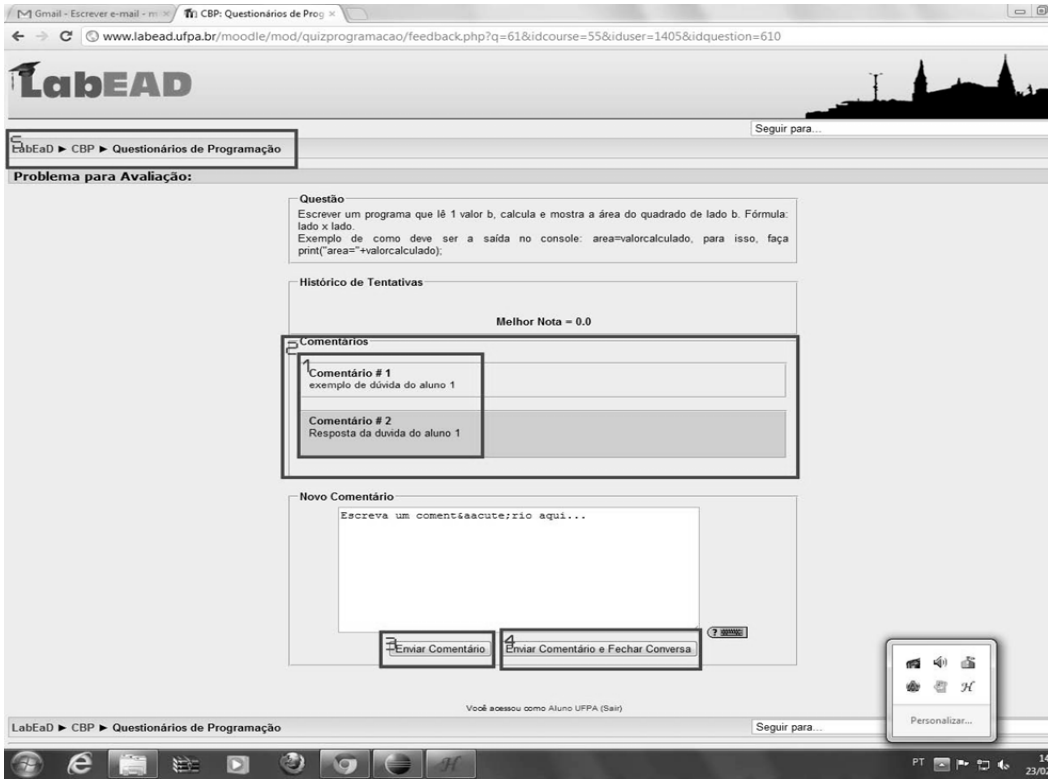
**Heurística:** Compatibilidade do sistema com o mundo real

**Heurística:** Consistência e padrões

**Heurística:** Reconhecimento ao invés de relembrança

**Grau de Severidade:** 3 -Problema de usabilidade menor

**Descrição do Problema**: Os títulos das "seções" não estão claros. O título “Ajudas pendentes” dá a entender que são ajudas que o estudante solicitou que ainda não foram atendidas, mas trata-se de ajudas que ele precisa responder. Deveria haver três seções: uma para as ajudas solicitadas por outros estudantes que o estudante precisa responder, uma para as ajudas solicitadas pelo estudante que ainda não receberam respostas e outra para as ajudas solicitadas do estudante já com resposta.




**Seleção 1**

**Heurística:** Reconhecimento ao invés de relembrança

**Grau de Severidade:** 3 -Problema de usabilidade menor

**Descrição do Problema:** Não há indicação de que estudante enviou qual comentário.

**Seleção 2**

**Heurística:** Controle do usuário e liberdade

**Heurística:** Consistência e padrões

**Grau de Severidade:** 3 -Problema de usabilidade menor

**Descrição do Problema:** Não existe a possibilidade de um estudante apagar um comentário seu (caso tenha errado, por exemplo).

**Seleção 3**

**Heurística:** Controle do usuário e liberdade

**Grau de Severidade:** 3 -Problema de usabilidade menor

**Descrição do Problema:** Quando se envia um comentário não existe nenhuma possibilidade na interface de fechar a tela de dúvidas e voltar ao questionário.

**Seleção 4**

**Heurística:** Visibilidade do status do sistema

**Heurística:** Prevenção de erros

**Grau de Severidade:** 4 -Problema de usabilidade grave

**Descrição do Problema:** *Label* do botão dá a entender que apenas a tela será fechada, mas o sistema finaliza a dúvida toda. Sugestão: substituir *label* para algo parecido com Enviar Comentário e Finalizar Ajuda.

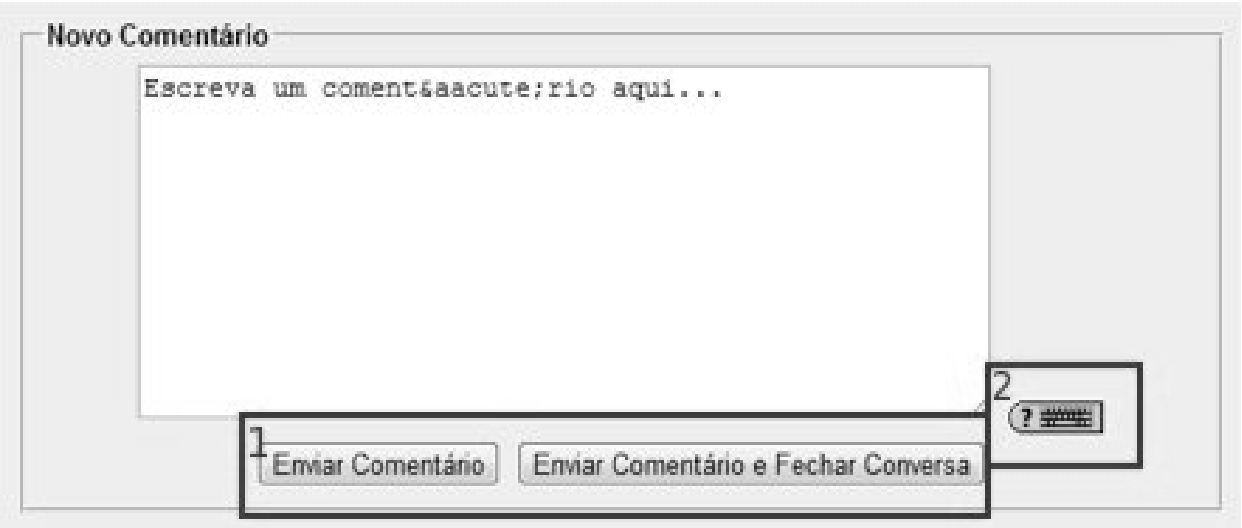
**Seleção 5**

**Heurística:** Controle do usuário e liberdade

**Heurística:** Consistência e padrões

**Grau de Severidade:** 4 -Problema de usabilidade grave

**Descrição do Problema:** O sistema não oferece a opção Cancelar ou Voltar. Não permitindo que o usuário saia da tela de ajuda senão pelo botão do próprio navegador. Não há a possibilidade de voltar à página do questionário caso, por exemplo, o estudante tenha acessado o link da dúvida errada. Só há a possibilidade de voltar à página principal do curso.




**Seleção 1**

**Heurística:** Help e documentação

**Grau de Severidade:** 3 -Problema de usabilidade menor

**Descrição do Problema:** O conteúdo da ajuda oferecida (os atalhos) não é tão importante para a caixa de texto “Novo comentário”. A ajuda deveria descrever o funcionamento da caixa e aí sim poderia existir um link para os atalhos.

**Seleção 1**

**Heurística:** Consistência e padrões

**Grau de Severidade:** 1 -Problema cosmético

**Descrição do Problema:** O conteúdo da ajuda oferecida (os atalhos) não é tão importante para a caixa de texto “Novo comentário”. A ajuda deveria descrever o funcionamento da caixa e aí sim poderia existir um link para os atalhos.

**Seleção 2**

**Heurística:** Consistência e padrões

**Grau de Severidade:** 3 -Problema de usabilidade menor

**Descrição do Problema:** O label do botão deveria estar apenas como “Enviar”, já que se sabe que é um novo comentário

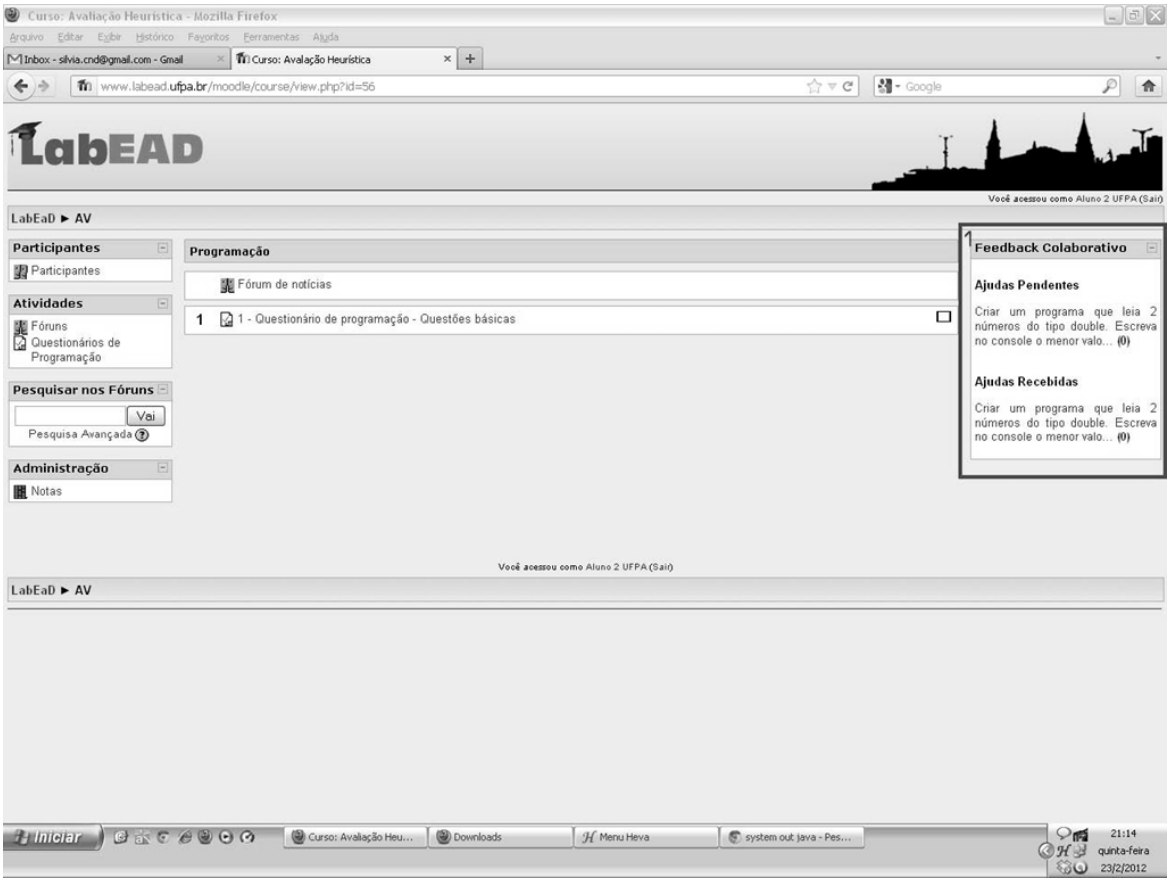
**Seleção 1**

**Heurística:** Compatibilidade do sistema com o mundo real

**Heurística:** Prevenção de erros

**Grau de Severidade:** 5 -Necessita correção imediata

**Descrição do Problema:** Supõe-se que o número entre parênteses deva indicar o número de ajudas pendentes e recebidas, no entanto este contador sempre está nulo. Isto pode fazer com que o usuário pense que não há nenhuma ajuda pendente, quando na verdade há.



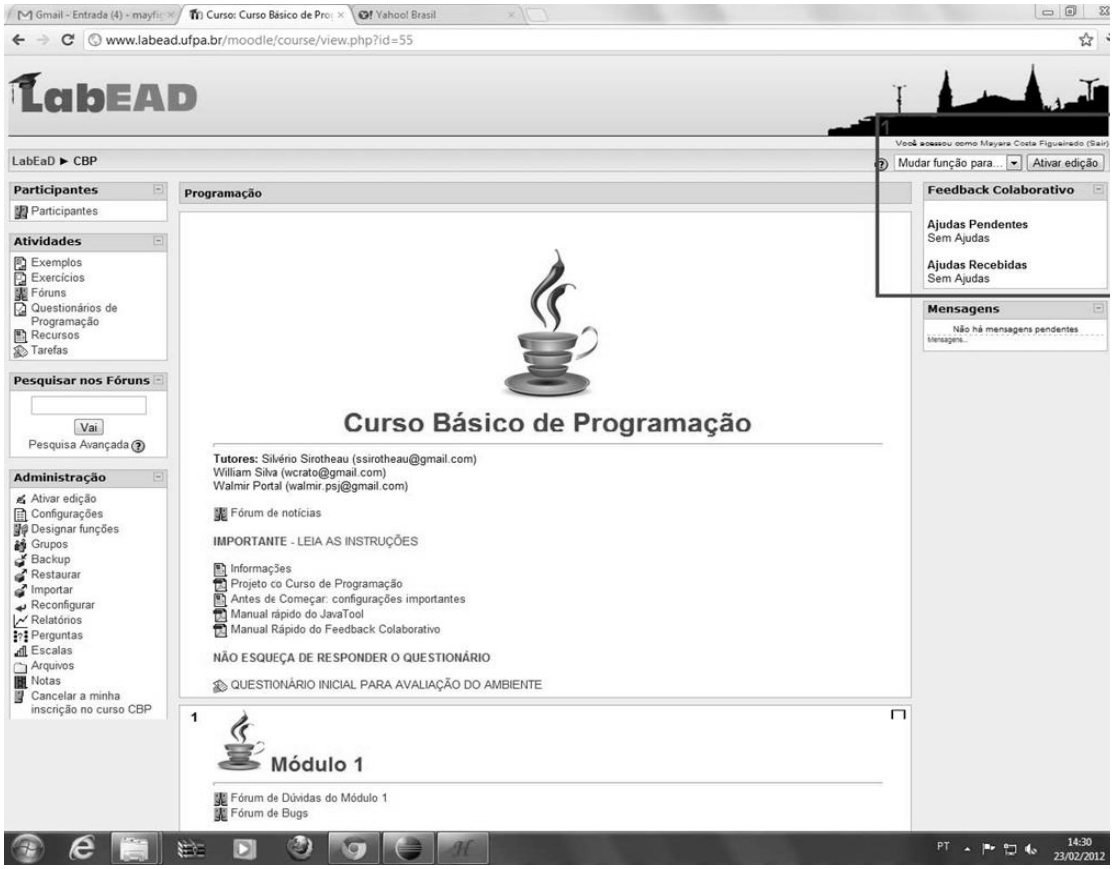

**Seleção 1**

**Heurística:** Compatibilidade do sistema com o mundo real

**Heurística:** Prevenção de erros

**Grau de Severidade: 5** - Necessita correção imediata

**Descrição do Problema:** Supõe-se que o número entre parênteses deva indicar o número de ajudas pendentes e recebidas, no entanto este contador sempre está nulo. Isto pode fazer com que o usuário pense que não há nenhuma ajuda pendente, quando na verdade há.



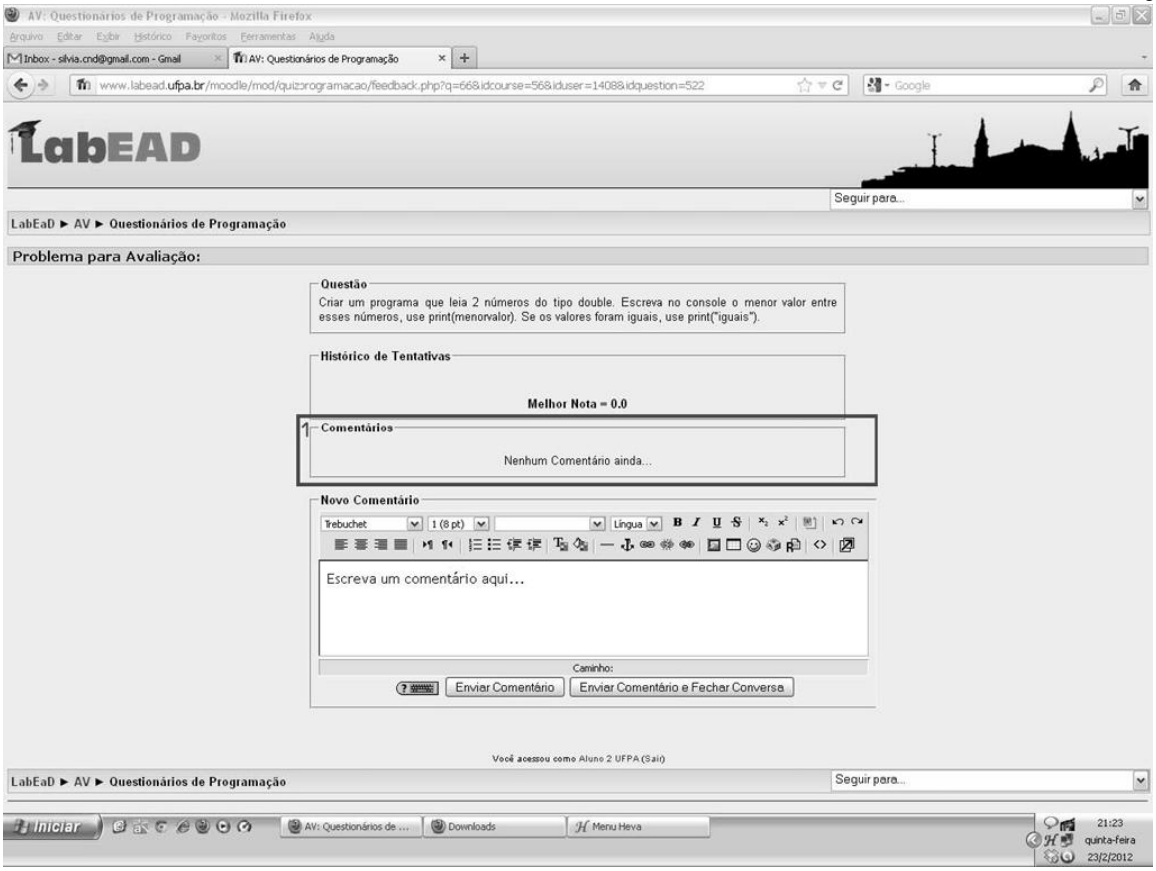

**Seleção 1**

**Heurística:** Consistência e padrões

**Heurística:** Reconhecimento ao invés de relembrança

**Grau de Severidade:** 3 -Problema de usabilidade menor

**Descrição do Problema:** Ao acessar com perfil de professor os itens selecionados não trazem nenhuma informação porque o acesso às dúvidas é feito através dos questionários. Entretanto, a presença da caixa selecionada dá a entender que não há nenhuma dúvida.

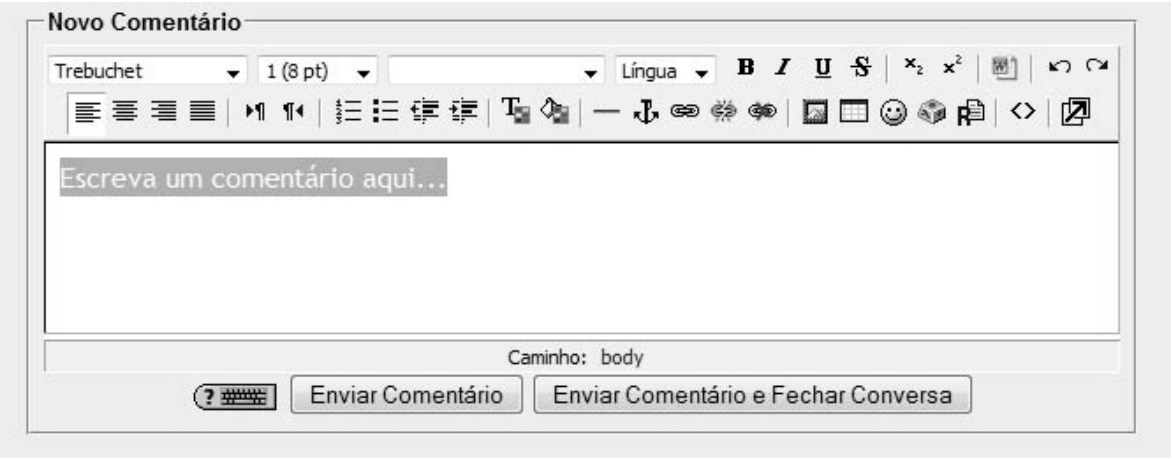



**Seleção 1**

**Heurística:** Visibilidade do status do sistema

**Grau de Severidade:** 5 -Necessita correção imediata

**Descrição do Problema:** Não sei se isso é um simples problema de usabilidade ou se é funcional. A questão é que ao clicar em "Ajudas Pendentes" o sistema redirecionou para uma tela vazia, ou seja, sem nenhum comentário. Uma vez que não aparece o problema a ser solucionado o estudante não tem como dar a solução.




## 5.2 Problemas de funcionamento em diferentes navegadores

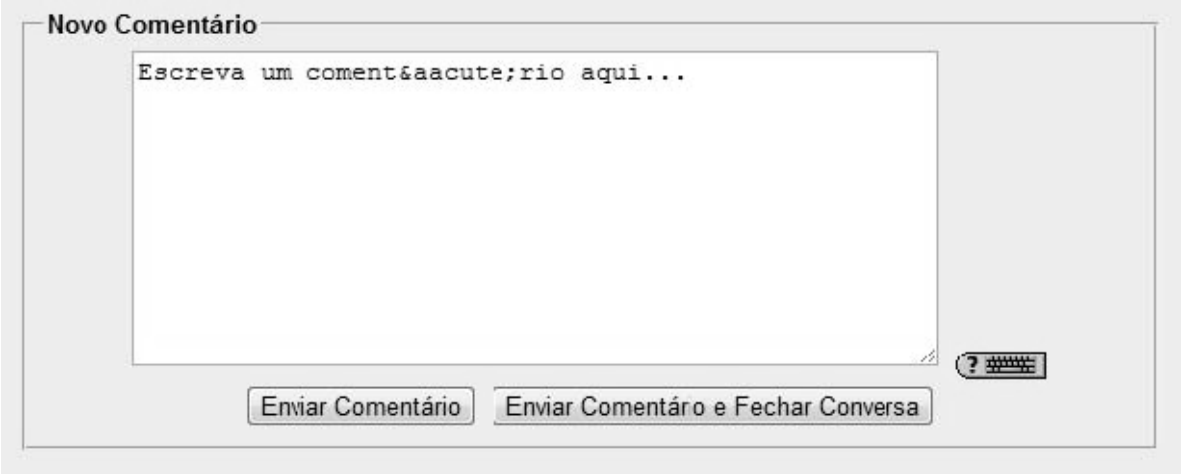


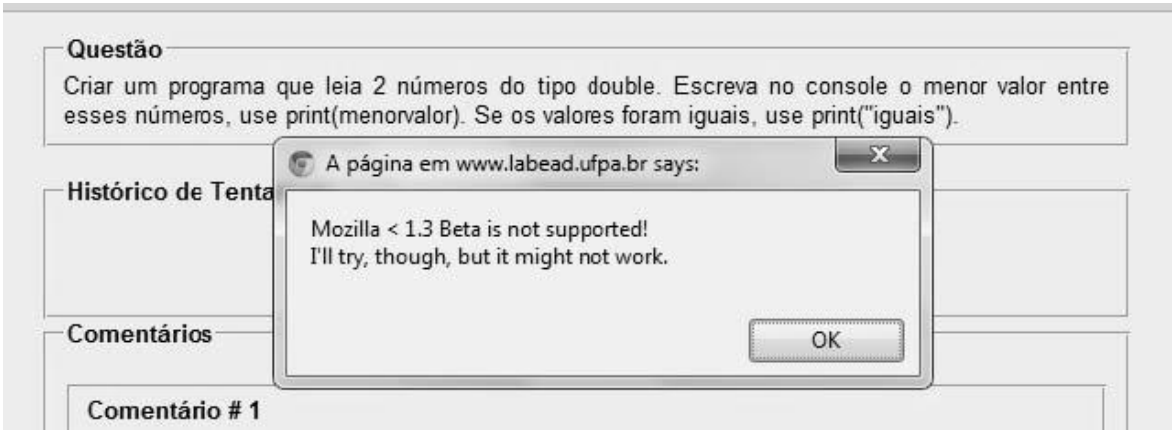

Imagem 9A -Caixa de texto no navegador Mozilla Firefox.

Imagem 9B -Caixa de texto no navegador Google Chrome

**Heurística:** Consistência e padrões

**Grau de Severidade:** 4 -Problema de usabilidade grave

**Descrição do Problema:** A caixa de texto se diferencia nos navegadores Mozilla Firefox 10 (Imagem 9A) e Google Chrome 17 (Imagem 9B). Considerando que são dois dos navegadores mais populares no mercado, recomenda-se que o sistema funcione igualmente nos dois navegadores.




**Heurística:** Consistência e padrões

**Heurística:** Ajudar os usuários a reconhecer, diagnosticar e corrigir erros

**Grau de Severidade:** 4 -Problema de usabilidade grave

**Descrição do Problema:** Ao utilizar o Google Chrome, a cada vez que é acessada a seção de ajuda, o usuário se depara com a seguinte mensagem de erro. Recomenda-se a adaptação do sistema para este navegador (Imagem 10) .A mensagem de erro está em inglês e não auxilia o usuário a entender o problema. Caso não seja possível adaptar o sistema, recomenda-se atualizar a mensagem de erro.

# **APÊNDICE A -** Questionário do Ensaio de Interação

1. A interface do Subsistema de *Feedback* é agradável?
2. A navegação no Subsistema de *Feedback* é fácil?
3. Os comandos do Subsistema de *Feedback* são claros?
4. Você encontrou dificuldades em realizar as tarefas da lista?
5. É fácil utilizar o Subsistema de *Feedback* Colaborativo?
6. Você ficou satisfeito com o Subsistema de *Feedback* Colaborativo?
7. Agora atribua uma nota de 0 a 10 ao Subsistema de *Feedback* Colaborativo.

**APÊNDICE B -** Termo de Consentimento Livre e Esclarecido

Você está sendo convidado(a) para participar, como voluntário, em uma pesquisa. Após ser esclarecido(a) sobre as informações a seguir, no caso de aceitar fazer parte do estudo, assine ao final deste documento, que está em duas vias. Uma delas é sua e a outra é do pesquisador responsável.

Desde logo fica garantido o sigilo das informações. Em caso de recusa você não será penalizado(a) de forma alguma.

**INFORMAÇÕES SOBRE A PESQUISA:**

Título do Projeto: **APRENDIZAGEM DE INICIANTES EM ALGORITMOS E PROGRAMAÇÃO:** FOCO NAS COMPETÊNCIAS DE AUTOAVALIAÇÃO

Pesquisador Responsável: SILVÉRIO SIROTHEAU

Telefone para contato: (91) 91484568

O objetivo é avaliar um grupo de usuários em um sistema de *feedback* colaborativo. Trata-se de um estudo, voltado para aprimorar o conhecimento autoavaliativa de estudantes de programação. A coleta será realizada em um laboratório de usabilidade, sendo que as variáveis analisadas foram através de uma lista de tarefas com 7 perguntas, com garantia de sigilo e direito de retirar o consentimento a qualquer tempo. Não há nenhum risco, prejuízo, desconforto ou lesões que podem ser provocados pela pesquisa.

- Nome e Assinatura do pesquisador:

- **CONSENTIMENTO DA PARTICIPAÇÃO DA PESSOA COMO SUJEITO**

Eu, _____, _____, abaixo assinado, concordo em participar do estudo _____ , como sujeito. Fui devidamente informado e esclarecido pelo pesquisador _____ sobre a pesquisa, os procedimentos nela envolvidos, assim como os possíveis riscos e benefícios decorrentes de minha participação. Foi-me garantido o sigilo das informações e que posso retirar meu consentimento a qualquer momento, sem que isto leve à qualquer penalidade ou interrupção de meu acompanhamento/ assistência/tratamento.

Local e data _____/_____/_____/_____/

Nome: _____

Assinatura do sujeito ou responsável: _____