Ensino e Aprendizagem de Programação: Análise da Aplicação de Proposta Metodológica Baseada no Sistema Personalizado de Ensino

Paulo Santana Rocha[1]

Universidade Federal do Pará (UFPA) – ICEN - paulo_srocha@hotmail.com

Benedito Ferreira[2]

Universidade Federal do Pará (UFPA) – ICEN - ferreira@ufpa.br

Dionne Monteiro[3]

Universidade Federal do Pará (UFPA) – ICEN - dionne@ufpa.br

Danielle da Silva Costa Nunes[4]

Universidade Federal do Pará (UFPA) – ICEN - danicostanunes@gmail.com

Hugo Cezar do Nascimento Góes[5]

Universidade Federal do Pará (UFPA) – ICEN - hugoescezar@gmail.com

Resumo. *Ensino e aprendizagem de programação tem sido foco de muitas pesquisas, nas quais se pode destacar uma ênfase significativa na questão das dificuldades discentes no entendimento de conceitos relacionados a esse aprendizado. Neste artigo será apresentada a concepção e implementação de um método de ensino baseado no Sistema Personalizado de Ensino, e de uma ferramenta integrada ao AVA Moodle, bem como uma análise de resultados de sua implantação em uma instituição de ensino. Os resultados obtidos, em confronto com o histórico de desempenho na disciplina, são bastante animadores para a continuidade da aplicação dessa metodologia.*

Palavras-chaves*: Informática Educativa, Metodologia de Ensino, Algoritmo, Sistema Personalizado de Ensino*

Abstract. *Teaching and learning of programming has been focus of much research, in which we can highlight a significant emphasis on the issue of students' difficulties in understanding concepts related to that learning. This paper will present the conception and implementation of a teaching method based on Personalized System of Education and a tool integrated into the Moodle LMS. An analysis of results of its implementation in an educational institution is also described. The results, in comparison with the performance*

history of the discipline, strongly encourages the continued application of this methodology.

Keywords: *Computers in Education, Teaching Methodology, Algorithm, Personalized System of Instruction*

1. Introdução

Em pesquisas realizadas nos últimos anos, como mostram Pereira e Rapkiewicz (2004), observa-se uma preocupação crescente com o processo de ensino/aprendizagem de programação. Tais estudos são motivados, sobretudo, pela importância dos conceitos de programação na vida acadêmica dos cursos de computação.

Existem várias possibilidades de origem das dificuldades no ensino de programação, como destaca Raabe e Silva (2005), seja pela exigência lógico-matemático predominante na disciplina, ou mesmo pela dificuldade de apreensão, por parte do professor, ritmo de aprendizagem de cada aluno. No que diz respeito ao ritmo de aprendizagem é importante destacar que o andamento da disciplina não necessariamente é condizido no ritmo de assimilação de cada aluno, fator que pode acarretar dificuldades na apropriação dos conceitos fundamentais, podendo ocorrer como conseqüência o desinteresse pelo conteúdo ministrado, ou até mesmo certa aversão à disciplina. É comum observarmos pesquisas apontando casos de evasão de curso, fato que tem relação íntima com as dificuldades de aprendizagem (Castro et al., 2003).

É importante ainda ressaltar enfaticamente que a apropriação ou não dos conceitos iniciais de programação tem relação direta com o desempenho do aluno no decorrer de todo o curso, já que disciplinas avançadas dependem fortemente desses conceitos.

Um fator limitante é a difícil tarefa do professor em identificar, em uma turma, as dificuldades individuais de cada aluno, tornando-se o aprendizado passível de falhas. A não identificação dos níveis de dificuldade dos educandos pode implicar em muitas dificuldades futuras, pois à medida que se iniciam atividades mais complexas envolvendo programação, ou conteúdos de alguma forma relacionados (modelagem de Software, Redes, Estrutura de Dados, etc.), a dependência de boa desenvoltura nos conceitos básicos aumenta significativamente.

Assim, um aspecto relevante é o ritmo de andamento do estudo: tipicamente, o conteúdo de uma disciplina de programação é apresentado aos alunos através de aulas expositivas ou práticas, onde o ritmo de evolução do conteúdo é dado pelo professor, tendo-se em vista o cumprimento de um conteúdo programático. Ocorre que esse ritmo não necessariamente se ajusta à condição de cada individuo. Alunos com dificuldades em assuntos tratados anteriormente tendem a não acompanhar o conteúdo em sua plenitude, o que pode ocasionar perda de assimilação e, igualmente importante, de interesse, fazendo com que os mesmos passem a ver a disciplina como algo aversivo e

preocupando-se essencialmente em conseguir a média mínima para seguir adiante no curso.

Adota-se neste estudo a hipótese de que um método que permita que cada aluno siga seu próprio ritmo de evolução nos conteúdos pode trazer melhores resultados, por se intervir diretamente nesse citado descompasso entre o nível alcançado pelo aluno e a exposição do professor, causador de desmotivação. Pretende-se, assim, neste artigo, descrever e analisar a aplicação de uma proposta metodológica baseada no Sistema Personalizado de Ensino, inspirada em Keller (1968), que permite uma abordagem perante o ensino de programação em que o aluno é parte integrante do processo de aprendizagem, passando a ser instrumento dinâmico e participativo. Espera-se, assim, garantir a cada estudante, independentemente de seu ritmo de avanço, iguais condições de assimilação dos conceitos e práticas necessários para um bom aprendizado de programação.

2. Lógica de programação nos cursos de computação: dificuldades no aprendizado.

Independentemente da metodologia e dos paradigma utilizado, lógica de programação é requisito fundamental nos cursos de computação, sendo instrumento importante na estruturação de raciocínio lógico e formulação de algoritmos corretos. No entanto, muitas vezes a condução desse estudo é visto com preocupação pelos alunos, sendo que mesmo antes do começo das atividades já se forma uma resistência perante o conteúdo, fato que pode ocorrer, segundo Pereira e Rapkiewicz (2004), por sua característica lógico-matemática, dado o enraizado estigma relacionado a esses conteúdos.

Procurando desmistificar a cultura de insucesso e reprovações criada sobre o ensino de programação, algumas pesquisas apontam como provável solução até a iniciação do aluno em lógica de programação ainda no nível médio (Pereira, et al., 2005), objetivando-se, além de estimular o aluno a optar por cursos na área de computação, dar condições de criação de soluções lógicas e matemáticas que podem ser utilizadas em seu cotidiano, independentemente da área de formação futura. No entanto, tem-se verificado que um dos grandes motivos de evasão escolar nos cursos de computação (Castro, et al., 2003), estão ligados às primeiras cobranças de raciocínio lógico, já que o aluno se sente incapaz de realizar tais abstrações lógicas. Raabe e Silva (2005), analisando as dificuldades existentes nesse campo de ensino, as classificam em três grandes grupos de natureza de problemas (didática, cognitiva e afetiva). Problemas que envolvem o ritmo de aprendizado do aluno, material utilizado, perfil comportamental ou mesmo problemas pessoais e afetivos são apontados como relevantes.

Para um melhor diagnóstico das dificuldades, é necessário um processo sólido de avaliação, uma vez que este é instrumento principal não somente para análise, mas, sobretudo, para tomada de ações efetivas sobre as dificuldades verificadas. Não havendo a devida intervenção a partir do diagnóstico das dificuldades do aluno, não se está praticando propriamente uma avaliação, mas uma condenação do aluno, como se só ele fosse responsável pelo fracasso. Esta cultura tão enraizada de reprovações coloca a

instituição de ensino em uma posição cômoda, onde se pode aprovar uns e reprovar outros, estando suposto que estes últimos são os "culpados" por seu infortúnio de não serem aprovados. (Paro, 2001)

3. Proposta Metodológica

O Personalized System of Instruction (PSI, traduzido para o português como Sistema Personalizado de Ensino) é uma metodologia ou sistema de ensino idealizado pelos professores Fred Keller, Carolina Martuscelli Bori, John Gilmour Sherman e Rodolpho Azzi em 1963, e embasada em princípios da Análise do Comportamento (Keller, 1968). No Brasil, foi implantado iniciamente na Universidade de Brasília em 1964, e amplamente utilizado, principalmente nos anos 70. Em 1979 havia, mundialmente, cerca de 5.000 cursos conhecidos baseados no PSI, um periódico específico para o assunto (Journal of Personalized Instruction) e, de 1973 a 1979, manteve-se em funcionamento o Center for Personalized Instruction na Georgetown University (Todorov et al, 2009)

A flexibilidade do método é uma de suas mais atraentes qualidades, pois possibilita o uso de uma diversidade de recursos educacionais, tradicionais ou não, para compor um curso no seu formato. O papel do professor no PSI diferencia-se de seu papel no ensino tradicional. A função principal deixa de ser a de transmitir conhecimento e passa a ser a de acompanhar, aprimorar, treinar e gerenciar (Todorov & Tristão, 1975).

Observando-se inúmeras citações na literatura sobre o ensino de programação e suas dificuldades, fica cada vez mais latente a necessidade de instrumentos e metodologias que possam dar condições iguais a todos de aprender os conceitos fundamentais, o que pode, segundo a hipótese desta pesquisa, ser obtido em uma abordagem em que cada educando possa seguir de acordo com seu ritmo próprio, abordagem típica do PSE. Esta metodologia foi aplicado em inúmeras áreas do saber, apresentando bons resultados práticos, como mostra Moreira (2004), no entanto, nunca aplicado ao ensino de programação.

Para que a aplicação da metodologia atinja seus objetivos, o PSI sustenta que alguns procedimentos devem ser adotados:

- **Disponibilização fácil de materiais e exercícios:** através de algum meio de comunicação (um ambiente virtual de aprendizagem, por exemplo) devem ser disponibilizados os instrumentos de estudos para os alunos, onde os mesmos possam ter acesso a qualquer momento para análise, estudo e resolução de problemas. Coerentemente com a proposta de *estudo assistido* desta metodologia, os materiais possuem um grau elevado de detalhamento e recorre-se bastante, para ilustrar os conceitos expostos, a animações que ilustram a execução do código, de forma concomitante à evolução das células de memória.
- **Organização em Níveis:** é necessário organizar os materiais e exercícios em níveis seqüenciais, permitindo aos estudantes seguir de forma contínua seu guia de estudos. Para tanto, um nível é pré-requisito para o subseqüente, existindo

entre um nível e outro um pequeno acréscimo de complexidade, possibilitando ao aluno avançar consistentemente de acordo com seu ritmo.

- **Mobilização de professores e monitores**: professores e monitores exercem papel fundamental na implantação da metodologia, tendo-se em vista que estes têm contato direto e contínuo com os alunos. Assim, poderão identificar as dificuldades mais comuns e intervir no sentido de sua superação (e, ainda, colher elementos para a melhoria da abordagem adotada). A relação ideal entre o número de alunos e professores/monitores difícil de ser rigidamente estabelecida; porém esta deve garantir que professores/monitores, apoiados pelas tecnologias disponíveis, tenham condições de acompanhar individualmente o avanço de cada aluno.
- **Disponibilização permanente a orientação e monitoramento:** como os materiais e exercícios estarão disponíveis a qualquer momento para os alunos, as dúvidas e dificuldades também poderão surgir na mesma freqüência, não necessariamente nos momentos agendados; assim é importante a disponibilização de recursos para comunicação com os alunos (correio eletrônico e fóruns, por exemplo).
- **Encontros periódicos de *estudo assistido*:** nos encontros periódicos com os alunos (aulas e encontros extra em laboratório) cada um deles, individualmente ou em pequenos grupos, prossegue seu estudo no nível onde se encontra. Nestes encontros, é fundamental a troca de experiências entre os colegas e a presença dos professores/monitores para tirar dúvidas.
- **Avaliação em cada nível:** a partir do acompanhamento por parte dos professores e monitores, os alunos poderão a qualquer tempo, sempre que se sentirem preparados, realizar uma avaliação referente àquele nível de estudo. Se demonstrarem proficiência naquele nível, deverão passar para o próximo nível de estudo.

Observe-se, como um ponto fundamental, que com a adoção de estudo por níveis, dado que cada aluno se encontra em um nível específico, não mais ocorre uma seqüencia de aulas expositivas, comuns a todos, mas um estudo assistido, conforme acima explicitado. No máximo, ações coletivas serão dirigidas aos grupos de alunos que se encontram em um mesmo nível.

Cabe destacar, ainda, que para a passagem de níveis, não se espera domínio absoluto dos conteúdos ou uma suposta perfeição, algo inclusive difícil de se definir. Entretanto, espera-se que o aluno atinja um grau de excelência, ou seja, não cometa erros fundamentais sobre o conteúdo daquele nível, e possua boa desenvoltura com seus conceitos. Diferentemente dos esquemas "tradicionais" de avaliação onde o aluno é considerado "aprovado" ao atingir 50% (em alguns casos, até 70%), aqui se propõe que para passar de nível, deve-se alcançar, em termos mensuráveis, algo na faixa de 90% a 100% de acertos. A nota final (ou conceito equivalente) atribuída na disciplina dependerá do nível que o aluno conseguir alcançar.

Para a experiência aqui relatada adota-se a proposta metodológica feita por Ferreira e Monteiro (2009), onde são definidos 14 níveis para estudo de programação imperativa. Como a aplicação se deu em uma disciplina denominada Algoritmos (disciplina inicial do curso Bacharelado em Sistemas de Informação, a ser complementada por outra no período subseqüente), adotou-se um roteiro de estudo com 6 níveis, conforme listado na Tabela 1[1].

Tabela 1: Relação de níveis para estudo de programação imperativa

Nível 01	Bases da programação imperativa
Nível 02	Estruturas de seleção: casos elementares
Nível 03	Estruturas de seleção aninhadas: situações de baixa/média complexidade
Nível 04	Estruturas de repetição I: a estrutura para (“laço contado”)
Nível 05	Estruturas de repetição II: estruturas enquantofaça e repitaaté.
Nível 06	Combinação de estruturas de repetição e seleção (baixa e média complexidade).

4. Proposta de Ferramenta computacional

Para a aplicação da metodologia, propõe-se uma ferramenta computacional para servir de apoio às atividades, tanto dos alunos como dos docentes/monitores. Tal ferramenta serve de repositório para materiais e exercícios, organizando-os de acordo com os níveis em cada módulo da disciplina, além da definição de fóruns de discussão sobre as dúvidas encontradas e sugestões dirigidas aos materiais didáticos, por nível de estudo. Do lado dos discentes, a ferramenta permite a submissão de soluções para os exercícios propostos, além da participação nos referidos fóruns, inclusive tirando dúvidas de colegas. Do lado dos docentes/monitores, há facilidades para o acompanhamento do desenvolvimento dos alunos, através da análise das suas soluções, tudo dentro do mesmo ambiente.

A atual aplicação da ferramenta proposta é desenvolvida como uma integração ao Ambiente Virtual de Aprendizagem Moodle (2010), um sistema amplamente utilizado e que possui vários recursos para apoio à aprendizagem, permitindo a postagem de arquivos, slides, apostilas e outros materiais, oferecendo também a possibilidade de inclusão de atividades, além de instrumentos de interação com os alunos, tais como: Fóruns, Chats, Tarefas, Lições, entre outros. Como se trata de um sistema de código fonte aberto, é possível implementar novas funcionalidades; assim, foi desenvolvido um novo recurso de organização dos materiais em níveis, onde os módulos estarão disponíveis a cada aluno à medida que este avance através da realização de um teste do nível estudado. Com isso se evita que os alunos dispersem seu

1 A proposta completa de formação básica em programação imperativa (que pode ser organizada em uma ou duas disciplinas, conforme a concepção curricular do curso) inclui ainda: Combinação de repetição e seleção com variados graus de complexidade; Vetores e matrizes; Programação modular; e Operações básicas sobre ponteiros.

esforço de estudo percorrendo materiais de níveis futuros, levando-os a se concentrarem no nível em que se encontram.

A Figura 1 ilustra uma tela da ferramenta adaptada à metodologia, onde se observa que os níveis 3 ao 10 estão desabilitados, o que significa que o aluno está estudando o nível 2 (a partir do momento que o mesmo realizar os testes deste nível, o subseqüente será liberado).

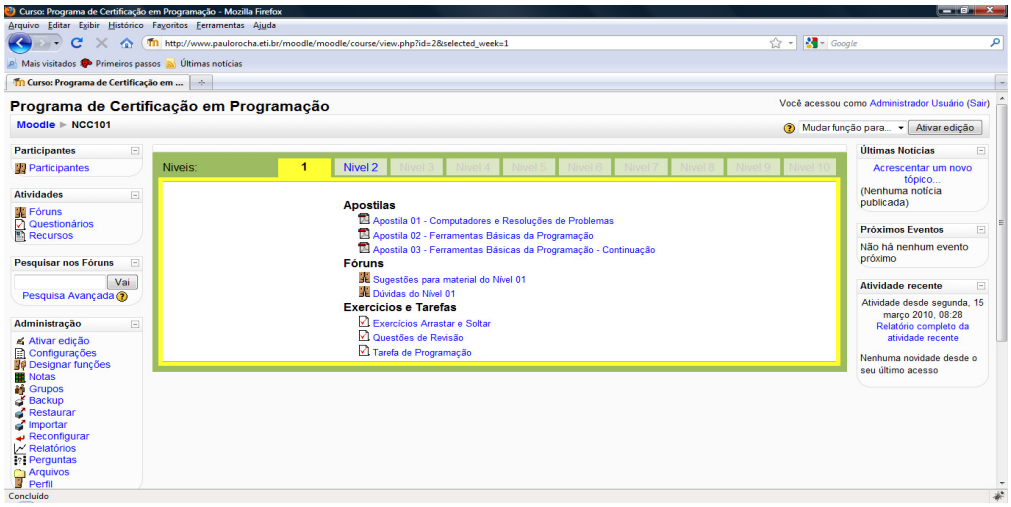


Figura 1. Moodle adaptado para aplicação da metodologia.

5. Estudo de Caso

Para análise da validade da metodologia e da ferramenta computacional proposta, realizou-se sua aplicação na disciplina Algoritmos, do primeiro semestre do Curso de Sistemas de Informação da Universidade Federal do Pará , com 32 alunos.

Adotou-se a linguagem Pascal para a resolução dos problemas (com ênfase não na linguagem, mas na lógica de programação), dada a conhecida adequação dessa linguagem ao ensino inicial de programação imperativa.

Os materiais (apostilas em forma de apresentação de slides com animações, exercícios e tarefas de programação) foram organizados em módulos, havendo entre um nível e outro um pequeno acréscimo de complexidade, de acordo com a concepção básica de que o aluno deve avançar através de pequenos – porém consistentes – passos.

Os encontros aconteciam sempre em laboratório, onde os alunos exploravam o material didático ao mesmo tempo em que testavam suas soluções nos compiladores. Levando-se em conta o ritmo individual de cada aluno, após a realização das tarefas e exercícios os mesmos se manifestavam sobre o momento adequado para realização das avaliações, que eram realizadas, a depender da demanda, semanalmente ou a cada duas semanas. Estas avaliações eram realizadas em papel ou no próprio ambiente computacional, com submissão do código, porém sem o teste da solução no compilador, para se evitar o método tentativa-e-erro e explorar a capacidade de abstração do aluno.

De acordo com as normas da instituição, deve-se atribuir a cada aluno um conceito no final do semestre. Nesta abordagem, como já citado, estes são atribuídos em função do nível alcançado, da seguinte maneira:

- Alunos que não alcançaram a conclusão do nível 04: conceito INS (insuficiente, que na instituição corresponde a uma média na faixa de 0 a 4,99);
- Alunos que concluírem o nível 04: conceito REG (regular, média na faixa de 5,0 a 6,99;
- Alunos que concluírem o nível 05: conceito BOM (bom, média na faixa de 7,0 a 8,99;
- Alunos que concluírem o nível 06: conceito EXC (excelente, média na faixa de 9,0 a 10,0).

Assim, o conceito final não se dá a partir de uma média das diversas notas. Uma conseqüência disso é que um aluno que tenha tido um mal desempenho na avaliação de um certo nível pode refazer essa avaliação e a nota anterior não terá efeito direto no conceito final, desde que ele consiga avançar subseqüentemente em seu estudo.

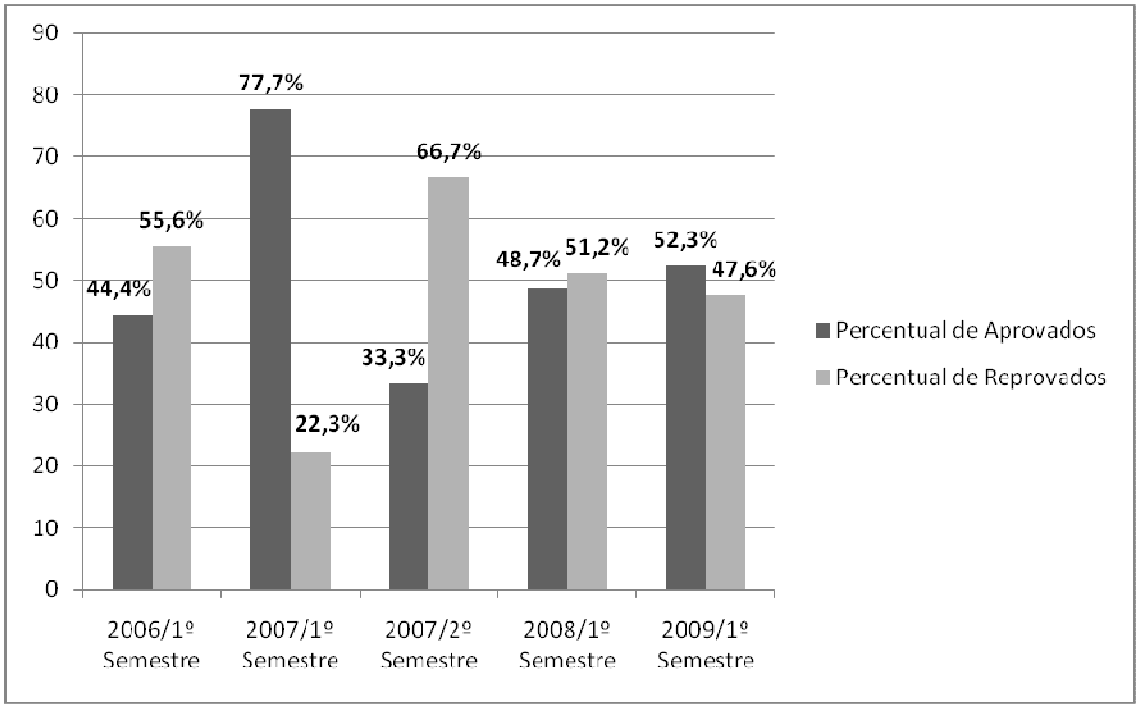
De posse do desempenho acadêmico (conceito final) da turma, e de questionários aplicados aos alunos, foram tabulados dados de forma a tornar possível uma análise da validade da aplicação da proposta metodológica, apresentada na próxima seção.

6. Resultados e Discussão

Procurou-se identificar, como parâmetro da eficácia da aplicação da metodologia, os percentuais de aprovação, evasão, e grau de autonomia na aprendizagem dos alunos participantes do projeto. Para fazer uma correlação dos dados obtidos, foi realizada uma análise histórica do aproveitamento nas disciplinas de Algoritmos e Programação da instituição de ensino[2]. Realizou-se um levantamento do desempenho discente no período de 2006 a 2009, em turmas ofertadas por diferentes professores. Como ilustra o Gráfico 1, houve um percentual elevado de reprovação na maioria dos semestres analisados: em todos eles, com exceção do primeiro semestre de 2007, as taxas de reprovação chegam próximo, e em muitos casos ultrapassaram 50%.

Gráfico 1: Desempenho discente - período de 2006/2009

2 Até dois anos atrás, havia no curso uma única disciplina básica de programação imperativa (*Programação*) de 90 horas. Atualmente, esta foi desdobrada em duas, *Algoritmos* e *Programação I*, ambas com 60h. Assim para efeito desta análise histórica, foram incluídas as ofertas tanto de *Programação* como de *Algoritmos*.



Considerando-se os conceitos obtidos pelos alunos aprovados (que podem ser REG, BOM ou EXC) constatou-se também que, dos alunos aprovados, mais de 45% obtiveram conceito Regular, a média mínima para aprovação. Esse fato denota certa deficiência no processo ensino/aprendizagem, podendo acarretar inúmeras dificuldades acadêmicas para o discente, já que, como dito anteriormente, os conceitos iniciais de programação são pré-requisito para disciplinas avançadas nos cursos de computação.

Considerando-se esse histórico, os dados obtidos com a aplicação da proposta metodológica constituem resultados bastante motivadores no que se refere aos percentuais e métricas analisadas. Quanto ao aspecto desistência ou evasão, no período de 2006 a 2009 essa taxa nas disciplinas Algoritmo e Programação girava em torno de 56%; já com a aplicação da presente metodologia, o percentual de evasão caiu para zero, o que evidencia uma boa intervenção no que diz respeito ao sentimento de aversão ao estudo, que interfere diretamente no abandono da disciplina.

Quanto ao desempenho discente na disciplina, obteve-se 84% de alunos aprovados, percentual bastante superior à média dos anos anteriores (em torno de 50%). Detalhando-se o desempenho dos alunos aprovados, constatou-se que, destes, 51,8% obtiveram conceito EXC e 25,9% alcançaram conceito BOM (portanto, apenas 22,3% de alunos com o conceito REG). Estes números indicam uma substancial melhora de desempenho coletivo, que reflete uma melhor apreensão do conteúdo da disciplina.

Outro aspecto analisado foi a capacidade de estudo autônomo dos alunos, já que, como citado anteriormente, não havia aulas no sentido tradicional. Nesse aspecto, todos os alunos afirmaram, através de formulários a eles submetidos no final da disciplina, que não tiveram dificuldades significativas para estudar e assimilar bem os conceitos utilizando o material disponível; mesmo aqueles que sentiram alguma dificuldade em

determinado módulo, após orientação do professor e monitores conseguiram avançar no seu estudo individual ou em pequenos grupos. Nesse sentido, entende-se que houve um reforço da concepção de que o aluno é parte integrante e fundamental do processo de aprendizado, não apenas um elemento passivo/receptivo.

7. Considerações Finais

Um primeiro aspecto a se destacar é que nos moldes tradicionais de ensino de programação imperativa a clara identificação das dificuldades de cada aluno é mais complexa: quando um aluno é reprovado, é mais difícil se visualizar em que pontos(s) houve deficiência na aprendizagem (de imediato, sabe-se apenas que na média das notas ele não alcançou o mínimo para aprovação), sendo portanto difícil realizar qualquer intervenção dirigida à situação particular de cada aluno. Em geral, ocorrerá a repetição da disciplina e de todo seu conteúdo, com chances significativas do aluno estar desmotivado e encontrar as mesmas dificuldades. Sobre este ponto, Paro (2001) afirma que o fracasso reiterado frustra e induz à aversão àquilo que o provoca; o aluno tem chances de progressivamente ver o ensino como algo penoso e aversivo.

Já no método ora descrito, mesmo com relação aos alunos reprovados na disciplina, conhece-se de imediato o grau de evolução alcançado, podendo-se orientar a continuidade do estudo a partir daquele ponto. Este aspecto dá ensejo a uma série de possibilidades de intervenção do ponto de vista de gestão acadêmica, como por exemplo: a) pode-se ofertar atividades nos períodos intervalares objetivando o atingimento, por parte do aluno, do nível necessário para passar à segunda disciplina de programação (Programação I), evitando-se tanto a reprovação e suas conseqüências negativas para o aluno, quanto o custo referente a reoferta da disciplina. b) mesmo após a realização (com aprovação) da série de disciplinas básicas de programação, é possível estimular a realização de estudos complementares, visando-se a obtenção de níveis próximos da excelência, com conseqüências positivas em diversas outras disciplinas do curso. Nesse caso, a concessão de créditos de atividades complementares pode ser um estímulo mobilizador.

Entende-se que a abordagem aqui descrita tem o potencial de tratar adequadamente alguns dos importantes problemas do ensino de programação, conforme apontam os resultados alcançados. E isto decorre, essencialmente da atenção especial que é dada ao ritmo individual de cada aluno, que tem relação com seu perfil pessoal, bem como com sua realidade social, uma vez que toda sua história escolar (virtudes e problemas de sua formação pregressa) interfere no aprendizado.

Como continuidade deste estudo, indicam-se as seguintes atividades: a) aplicar a metodologia na segunda disciplina de programação do curso (processo já em fase de planejamento); b) incentivar a aplicação da metodologia em outras instituições de ensino, para uma análise mais abrangente de sua eficácia; c) implementar funcionalidades adicionais na ferramenta computacional, como correção automática dos programas, geração automática de provas, dentre outras.

8. Referências

Cunha, M. V. (2000) "Psicologia da Educação", Editora DP&A.

Castro, C. T., Castro Junior, A., Meneses, C., B., M. e Rauber, M. (2003) "Utilizando Programação Funcional em Disciplinas Introdutórias de Computação", In: XI Workshop de Educação em Computação – WEI, Campinas/SP.

Ferreira, B.; Monteiro, D. (2009) "Projeto de Ensino – Programa de Formação e Certificação em Fundamentos de Programação Imperativa", Disponível em http://www.aedmoodle.ufpa.br/moodle/file.php/133/ProjetoCertificProgramacao.pdf. Acesso em julho de 2010.

Keller, F. (1968) "Good-Bye Teacher", Journal of Applied Behaviour Analyses, No1.

Moodle (2010) "A Free, Open Source Course Management System for Online Learning". Disponível em http://moodle.org. Acessado em março/2010.

Moreira, M. B. (2004) "Em casa de ferreiro, espeto de pau: o ensino de Análise Experimental do Comportamento", in: Revista Brasileira de Terapia Comportamental e Cognitiva.

Paro, V. H. (2001) "Reprovação Escolar: renúncia à educação", 2ª Edição, Editora Xamã, São Paulo.

Pereira Junior, J.C.R., Rapkiewicz, C. (2004) "O Processo de Ensino-Aprendizagem de Fundamentos de Programação: Uma Visão Crítica da Pesquisa no Brasil", WEI RJES.

Pereira Junior, J.C.R., Rapkiewicz, C., Delgado, C., Xexéo, J.A.M. (2005) "Ensino de Algoritmos e Programação: Uma Experiência no Nível Médio", comunicação particular, março, Rio de Janeiro.

Raabe, A. L. A., Silva, J. M. C., (2005) "Um Ambiente para Atendimento as Dificuldades de Aprendizagem de Algoritmos". XXV Congresso da Sociedade Brasileira de Computação. São Leopoldo/RS.

Todorov, J. C., Moreira, M. B., Martone, R. C., (2009) "Sistema Personalizado de Ensino, Educação a Distância e Aprendizagem Centrada no Aluno". Psicologia: Teoria e Pesquisa. São Paulo/SP.

Todorov, J. C., Tristão, G., (1975) "Sistema personalizado de ensino: bases psicológicas e abordagem administrativa" Cadernos de Psicologia Aplicada, 3, 65-71. São Paulo.