Uma Abordagem Pedagógica para a Iniciação ao Estudo de Algoritmos

Carla Delgado[1], José Antonio Moreira Xexeo[1], Isabel Fernandes de Souza[1], Marcio Campos[1], Clevi Elena Rapkiewicz[2]

[1]Faculdades Integradas Bennett, Rio de Janeiro – RJ

[2]Laboratório de Engenharia de Produção – Universidade Estadual do Norte Fluminense (UENF), Campos – RJ

delgado@pobox.com, xexeo@bennett.br, isabelfs@ig.com.br, camposmf@aol.com, clevi@uenf.br

Abstract: This paper describes a learning methodology for introductory disciplines on computer science undergraduate courses and reports the process of its implementation by a team of professors. The methodology is based on the role of faculties as mediators, on the pro-active attitude of the students and on the use of didatic resources to support the development of abstraction capacity, logic reasoning, problem solving skills and cognitive autonomy.

Resumo: Este artigo apresenta uma metodologia de aprendizagem para as disciplinas introdutórias de algoritmos e programação, nos cursos superiores de computação, e relata a experiência de um grupo de professores na sua concepção e implantação. A metodologia baseia-se na atuação do professor como facilitador, no papel pró-ativo do aluno e em recursos didáticos voltados para o desenvolvimento da capacidade de abstração, do raciocínio lógico, da solução de problemas e da autonomia cognitiva.

1. Introdução

As novas referências estabelecidas pela era da informação e a introdução progressiva da tecnologia na área educacional modificaram a relação entre aluno e professor, principalmente no ensino superior. Até pouco tempo atribuía-se ao mestre a maior parcela de responsabilidade na transmissão, aos seus alunos, de um conhecimento estratificado e, muitas vezes, ultrapassado. Hoje, o aluno transformou-se no foco do processo. Sua relação com o professor é pró-ativa, apoiada na facilidade de acesso à informação, mas dificultada pela instabilidade e pelas contradições dos novos modos de conhecimento e regulação social. Ao professor cabe o papel de facilitador no desenvolvimento da aprendizagem, para o qual se exige qualificação acadêmica, experiência profissional e capacidade para interagir com seus alunos e ajudá-los a interpretar as mudanças com as quais convivem.

A área de computação é constituída por tecnologias novas, em fase de grande expansão, contínuas modificações e estágios de maturidade heterogêneos. Essa situação gera um desafio aos professores; suas técnicas de ensino ainda não estão maduras e, portanto, sofrem adaptações freqüentes. Por exemplo, construir algoritmos e transformá-los em programas é, em alguns casos, um verdadeiro processo de alfabetização. São

novas linguagens que permitem formalizar soluções sistemáticas de problemas e o uso de máquinas para implementá-las, desde que sejam apreendidas e utilizadas com desembaraço e proficiência. Outro desafio – anterior e no mesmo patamar de dificuldade - é a própria elaboração da solução do problema enfrentado (Levitin 2000). É necessário construir um modelo do mundo real que propicie a tradução para a linguagem computacional. Desenvolver essa capacidade também é uma discussão em aberto.

Assim, como no processo de alfabetização, a formalização ou a expressão de situações e procedimentos, objetivos do estudo da computação, cria diversas dificuldades, como por exemplo:

- manipulação e interpretação de uma nova representação semiótica;
- identificação dos elementos, e de suas inter-relações, de uma situação problema;
- adaptação ao pensamento algorítmico.

Esse contexto dificulta a adaptação, ao ambiente universitário, dos alunos dos primeiros períodos dos cursos da área de computação; se não for considerado no momento da concepção das práticas da sala de aula, prejudica o rendimento acadêmico e a identificação com a futura profissão.

As dificuldades afloram principalmente no transcorrer das disciplinas ligadas ao estudo de algoritmos e programação e consideradas essenciais em todos os currículos de cursos de graduação em computação, tais como: Introdução à Ciência da Computação, Lógica Matemática e Algoritmos e Programação.

Segundo Chaves de Castro *et al* 2003, os estudantes encontram um obstáculo muito grande em aplicar suas habilidades prévias, criando uma fonte de medo e frustração. As conseqüências diretas são reprovações sistemáticas, apatia, baixa autoestima, desistência da disciplina, culminando com o abandono do curso.

Vários grupos de pesquisa, Rosso & Daniele (2000), Suraweera (2001), Chaves de Castro *et al* (2003) sugerem minimizar esses efeitos implantando, no ambiente escolar, técnicas mais adequadas para a aprendizagem desses assuntos. A de Carvalheiro & Setzer (1995) foi reproduzida com sucesso. Há outras iniciativas que desenvolvem ferramentas computacionais para potencializar os instrumentos de ensino já utilizados. Porém, ferramenta sem apoio de uma metodologia de ensino adequada traz poucos resultados para os propósitos assinalados.

Um dos pontos discutidos é a introdução ou não do paradigma de orientação a objetos no curso introdutório de algoritmos. O juízo de alguns grupos de pesquisa, tais como Marion (1999), Perego (2002) e Burton & Bruhn (2003) é de discordância; não há evidências significativas de que a nova metodologia facilita o aprendizado nos cursos iniciais sobre algoritmos. A abordagem imperativa estruturada clássica ainda é legítima para tratar vários tipos de problemas, além de ter a vantagem de incorporar diversos conceitos importantes de forma menos elaborada do que em outras abordagens.

Este trabalho está inserido no contexto de um projeto pedagógico que incentiva a experimentação e o teste de novas abordagens de aprendizagem e integra várias estratégias de ensino, potencializando a grade curricular e o ensino da computação (Santos *et al* 2003).

Nosso objetivo é apresentar uma metodologia de aprendizagem para as disciplinas introdutórias de algoritmos e programação, desenvolvida e adotada por um grupo de professores das Faculdades Integradas Bennett (FIB) do Rio de Janeiro. A metodologia é baseada na atuação do professor como facilitador[1], no papel pró-ativo do aluno e em recursos didáticos voltados para o desenvolvimento da capacidade de abstração, do raciocínio lógico, da solução de problemas e da autonomia cognitiva.

As discussões entre os professores que participam do projeto e a análise de trabalhos correlatos levaram ao estabelecimento de alguns pré-requisitos para a metodologia, a saber:

- o ritmo de trabalho de cada aluno e o acompanhamento correspondente é individual;
- a carga e a complexidade de trabalho são progressivas e não uniformes, sendo intensificadas ao longo do curso;
- o conhecimento é adquirido através da resolução de exercícios, em um processo de reflexão e descoberta.

2. Agentes

2.1. Alunos

O curso de Ciência da Computação das FIB é um curso noturno inserido no ambiente de uma instituição particular de ensino superior. Os alunos dos primeiros períodos formam um grupo heterogêneo, desde recém formados no ensino médio, até profissionais experientes já inseridos no mercado de trabalho, inclusive de computação. A maioria estudou em escolas particulares, a totalidade possui computador em casa e todos têm acesso à Internet.

Esse perfil não justifica o baixo aproveitamento nas disciplinas em pauta. Segundo Giraffa *et al* (2003), as causas principais envolvem os hábitos de estudo centrados em memorização e a falta de pré-requisitos em conteúdos correlatos, principalmente nos conteúdos de Matemática. A pesquisa de Falkembach *et al* (2002) apontou que os óbices no desenvolvimento de algoritmos advém da dificuldade de resolver problemas, de interpretar enunciados, de abstrair e formalizar informações.

A partir dessas constatações, fica evidente a necessidade de abordar interpretação, formalização e resolução de problemas, antes da apresentação dos conteúdos que fazem uso destas habilidades. Compreender a solução algorítmica dada a um problema, com ou sem o auxílio de instrumentos pedagógicos computacionais, é um passo posterior na questão de resolução de problemas - ciência (ou arte?) muito anterior ao computador.

2.2. Professores

Os professores envolvidos com o projeto são pós-graduados, contratados em regime de tempo contínuo, com formação básica na área de computação, visão do conjunto do

[1] Professor que incentiva, orienta e media seu(s) aluno(s) para a participação e na condução da aprendizagem. O professor facilitador forma uma parceria com o aluno para efetivar a aprendizagem, não se limitando a expor conteúdos.

curso, formação ou interesse na área pedagógica aplicada à computação e consciência da necessidade de atuar de forma transformadora nos processos de aprendizagem e de evolução da postura do discente como universitário. Portanto, com perfil adequado para realizar a experiência proposta.

Entretanto, a qualificação profissional docente não é suficiente. É certo que os professores entrevistados na pesquisa de Falkembach *et al* (2002) têm perfil adequado. Mas a pesquisa registrou que os professores de algoritmos - pelo menos os que foram entrevistados - não adotam estratégias centradas na resolução de problemas, nem se preocupam com o processo de formalização; a teoria é mostrada, basicamente, a partir de exemplos de algoritmos. McKeown e Farrell (1999) e Chaves de Castro *et al* (2003) também ressaltam que os cursos introdutórios raramente exploram técnicas de solução de problemas.

Portanto, além do perfil adequado, os professores precisam ter outros atributos importantes para participar de projetos desse tipo, nos quais o ambiente é de mudança. Elegemos alguns no trabalho de Brito Prado (2004): admitir as singularidades das situações, saber conviver com a incerteza, aprender a construir e a comparar novas estratégias de ações, novas teorias, novos modos de enfrentar e de definir os problemas.

A nossa experiência sugere ainda que, além desses atributos, o sucesso do projeto depende de outros pré-requisitos: existência de uma equipe de professores com tempo disponível para discutir e elaborar as técnicas e os procedimentos requeridos pelo processo, compromisso desses professores com a metodologia proposta, com o novo papel de facilitador e com a aprendizagem dos alunos. Afinal, o resultado obtido nas primeiras disciplinas de Algoritmos e Programação repercute ao longo de todo o curso.

3. O Processo

Ensinar a construir algoritmo talvez seja um objetivo difícil de realizar. A nossa experiência conduz à atitude de motivar, estimular e orientar o aluno a conquistar o seu próprio conhecimento, desenvolver as suas próprias técnicas e estratégias e obter soluções de sua autoria, em lugar de tentar absorver aquelas organizadas e apresentadas pelo professor.

A metodologia proposta é constituída por três fases: a primeira trabalha a resolução dos problemas[2], valorizando a autonomia cognitiva. Em seguida, o professor conduz o aluno pela experiência da formalização, valorizando a concisão e a precisão da linguagem utilizada. Finalmente, o objetivo passa a ser a construção de algoritmos.

O aluno é o agente principal na consecução dessas fases. O recurso didático essencial é a solução de problemas. O enfoque é o seu progresso nos aspectos de capacidade de abstração, do raciocínio lógico, da solução de problemas e da autonomia cognitiva.

---

[2] A importância da habilidade de resolução de problemas no processo de desenvolvimento de algoritmos foi tratada em Kortright 1994, Suraweera 2001, Rosso & Daniele 2000.

3.1. As Fases

3.1.1. Primeira Fase – Iniciação Lúdica – Resolução de Problemas

A primeira fase não trata de algoritmos ou de programação e sim de resolução de problemas de diversos domínios. O objetivo é desenvolver a autonomia na busca de soluções próprias para os problemas apresentados. Para isso, a imagem de que há necessidade de conhecimento prévio de um método específico de resolução é desconstruída, estabelecendo a validade do processo de tentativa e erro (Levitin, 2000) – evidentemente não aleatório - na busca de soluções.

O professor age como facilitador das atividades, sugerindo exercícios para explicitar questões despercebidas e indicando formas alternativas de interpretar o problema e encaminhar as tentativas de solução. Sugere também material suplementar para consulta e propõe formação de pequenos grupos para discussão de temas com entendimento ainda superficial. Muitas teorias de aprendizagem hoje utilizadas são categóricas ao afirmar que a aquisição do conhecimento só ocorre com a experiência, é o aprender fazendo (Chaves de Castro *et al*, 2003).

Os problemas propostos são de diversos domínios, não requerem conhecimentos específicos e não são relacionados à programação. Este universo inclui quebra-cabeças simbólicos, quebra-cabeças lógicos, jogos e charadas, problemas simples de aritmética e geometria.

Esta fase visa a capacitação do aluno, principalmente no processo de percepção e interpretação do problema e no hábito de analisar desafios de diferentes domínios. A fase culmina na auto-afirmação do aluno para ponderar e validar suas próprias soluções.

3.1.2. Segunda fase – Falando Sério - Formalização

A segunda fase - formalização do método utilizado para a solução – é trabalhada em duas etapas.

Na primeira dessas etapas, o objetivo é a formalização em linguagem natural. Cada aluno expõe a sua solução, verbalmente, a um grupo de colegas. Simultaneamente, o grupo é estimulado pelo professor a explorar, questionar e validar a solução apresentada pelo colega. Essa verbalização constrói, sob a intervenção dos componentes da turma, progressiva e interativamente, a formalização considerada satisfatória pelo grupo.

O papel do professor nessa etapa é o de minimizar o nível de competitividade e manter o grupo em ação colaborativa e investigativa.

Valoriza-se a discussão em grupo como um instrumento para despertar a percepção das fragilidades da linguagem natural. Lacunas intrínsecas ao processo de verbalização de soluções, corretas ou não, são percebidas na discussão, evidenciando as diferenças entre relatar, sintetizar e formalizar.

Por ser interativa e não documentada, a primeira etapa não comporta problemas que pressuponham soluções extensas e elaboradas.

A segunda etapa visa a autonomia no julgamento do nível de formalização adequado para a audiência; neste ponto inicia-se a conscientização de que existem diferentes níveis, e que a máquina exige o maior nível.

A etapa começa com a transcrição da solução de um aluno no quadro negro. Dessa forma são acrescentados outros elementos cognitivos, tais como sinais gráficos e fórmulas matemáticas, aumentando a gama de recursos disponíveis para a formalização. O aluno conclui sua transcrição, sem interferência, quando julga que a descrição formal está adequada. Este primeiro passo não é, portanto, interativo.

Concluída a transcrição, surge, espontaneamente, um debate sobre o que foi transcrito, gerando correções ou outras propostas de solução. Ao identificar alterações ou propostas novas consistentes, o professor determina a suspensão da discussão e a transcrição da respectiva intervenção, num processo cíclico de aperfeiçoamento.

A documentação (uso do quadro negro como memória) propicia o emprego de recursos visuais e referências cruzadas, o que permite o tratamento de problemas mais complexos. Esta evolução se refere aos instrumentos de formalização. Evolução mais importante é a conquista da autonomia; fica a cargo do autor a decisão sobre o nível de formalismo adequado para a transcrição no quadro negro. A redução das dificuldades identificadas na etapa anterior é um indicador da qualidade do processo de formalização apresentado.

3.1.3. Terceira fase – E agora? – Construção de Algoritmos

Finalmente, a construção do algoritmo! A linguagem de expressão não tem modelo, nem é imposta. O ciclo de aperfeiçoamento do processo de formalização induz, progressivamente, à criação de diversas estruturas simbólicas validadas pela turma para a comunicação escrita entre os alunos. O papel do professor, nesse ponto, é induzir a adoção de estruturas próximas daquelas utilizadas nas linguagens de programação imperativas.

Depois de formalizar, o aluno deve focalizar o processo de solução e não mais a solução em si. O objetivo é transformar a formalização desenvolvida em procedimentos sistemáticos que possam ser seguidos por outra pessoa, desde que qualificada para utilizá-los.

A passagem da segunda para a terceira fase não é pontual e sim gradativa. Os problemas apresentados aos alunos passam a exigir soluções mais elaboradas, nas quais cada vez mais é inerente o ato de explicitar procedimentos. A linguagem vai melhorando, as expressões das soluções vão se refinando, até o ponto em que são apresentados os conceitos de algoritmo e programa.

A apresentação desses conceitos justifica para os alunos a importância de se construir procedimentos; a descoberta da solução é mero requisito. Quanto mais consciente dessa importância, mais motivado fica o aluno para aumentar sua proficiência.

No decorrer deste processo, os alunos são instados à descoberta de técnicas rudimentares como divisão em casos, minimização e refinamentos, na medida que elas estejam ao alcance do aprendizado dos alunos. Não há pressa, em compromisso em apresentar estes conteúdos no curso introdutório. A necessidade do aprendizado deve partir naturalmente do envolvimento consciente (não mecânico) dos alunos com suas atividades práticas.

3.2. As Aulas

Existem três tipos de aula, que vão se alternado ao longo do curso: aulas de introdução, aulas de sincronização e aulas práticas.

3.2.1. Aulas de Introdução

As aulas introdutórias visam a despertar o interesse dos alunos para os desafios de uma nova fase, convencendo-os das possibilidades que podem ser exploradas e provocando sua curiosidade. Ao concluir a apresentação, o professor indica o material didático correspondente e orienta os alunos sobre o modo de trabalho.

3.2.2. Aulas de Sincronização

O objetivo das aulas de sincronismo é reduzir os efeitos das diferenças dos níveis de aprendizado já alcançados pelos alunos. O professor, que os acompanhou durante este período, faz uma revisão daqueles aspectos nos quais constatou falta de desenvoltura e também das descobertas mais significativas.

Esta aula abre espaço para que os alunos participem, contribuindo com suas experiências, falando dos pontos que acharam interessantes, das dúvidas que tiveram e das conclusões a que chegaram.

3.3.3. Práticas

As aulas práticas, maioria em todas as fases, visam a oferecer ao aluno um espaço – tempo para explorar os problemas nos quais está trabalhando, seja utilizando o computador ou outro material didático como livros e listas de exercício.

Este é um momento de trabalho individual ou em grupo, com o professor no papel de facilitador, orientando os alunos na indicação de referências e exercícios, propondo o agrupamento de alguns alunos que estejam desenvolvendo atividades em comum e auxiliando quando solicitado.

Para os alunos com maior velocidade de aprendizado, o professor propõe tarefas desafiadoras. Para os de menor velocidade, cabe o incentivo e mais rigidez na condução do aprendizado, de forma a manter o aluno o mais próximo possível do ritmo médio da turma.

3.4. Material Didático

3.4.1. Listas de exercícios

O material didático central da metodologia consiste em uma gama de exercícios que visam ao desenvolvimento das habilidades discutidas nas três fases do processo, acima descritas. Os exercícios têm níveis gradativos de dificuldade e são apresentados num contexto de desafio, fazendo o aluno passar, naturalmente, pela seqüência: percepção, motivação, reflexão, descoberta, questionamento, aprimoramento e sedimentação de novos conhecimentos.

O aluno que precisar de auxílio é atendido com a proposta de problemas complementares com orientação específica para a superação da dificuldade detectada. Os exercícios são organizados com dois objetivos: o primeiro é explicitar, separadamente, todos os segmentos dos processos de solução correspondentes à

dificuldade original. O segundo refere-se à organização desses segmentos como passos intermediários na solução do problema. As listas complementares exploram exaustivamente o tema em questão: todas as nuances que possam passar despercebidas devem ser evidenciadas.

O processo de elaborar, selecionar e organizar os exercícios é um processo longo e trabalhoso. No melhor cenário, deve ser feita por um grupo de professores envolvidos com o ensino das disciplinas introdutórias, através da pesquisa de problemas com as características desejadas e analisando os efeitos provocados nos alunos. Como o perfil dos alunos evolui constantemente, os efeitos provocados também se modificam e o material deve evoluir para continuar efetivo. Isto não é exatamente uma novidade na área de ensino de computação.

3.4.2. Referências bibliográficas

É muito importante disponibilizar, para o aluno, referências bibliográficas e "webliográficas". Uma vez que a autonomia, mesmo relativa, sobre o próprio aprendizado é alcançada, o aluno passa a buscar outras fontes de conhecimento, movido pela curiosidade ou pela reação crítica ao que lhe é apresentado, natural no processo. A autonomia, flexibilidade e a liberdade evolutiva são princípios básicos da metodologia. Esta iniciativa do aluno é excelente indicativo e deve ser incentivada e facilitada pelo professor e pelo ambiente (que deve disponibilizar os recursos necessários).

Uma pequena amostra de referências interessantes:

- Livros: Polya (1957), Magri (2003), Souza (2002).
- Links
  - problemas interativos: www.windesigner.hpg.ig.com.br/logica.htm
  - apostilas da área de computação: www.apostilando.com

3.4.3. Lista de atividades práticas de laboratório

Os objetivos almejados nesta metodologia independem do uso do laboratório. Entretanto, sua utilização, associada a uma linguagem de programação é motivadora. Neste contexto, o laboratório representa apenas mais um ambiente para a realização de aulas práticas.

As atividades propostas para uma aula prática em laboratório devem explorar os recursos computacionais disponíveis sem desviar o foco dos objetivos da fase que está sendo trabalhada. Dentre as possibilidades disponíveis estão teste, leitura, avaliação e construção de programas a partir de algoritmos feitos pelo professor ou pelo próprio aluno. Alguns grupos de pesquisa valorizam a utilização de ferramentas específicas de auxílio à construção de algoritmos. Os benefícios do emprego de instrumental deste tipo serão potencializados a partir do bom desempenho do aluno na terceira fase.

Uma atividade prática de laboratório é planejada como uma seqüência de tarefas coesas a serem realizadas integralmente durante o tempo de uma aula. Este pressuposto se justifica dada a pouca intimidade dos alunos com o computador como instrumento de trabalho e não de lazer. Uma postura menos objetiva em relação ao uso do computador pode levar à perda do foco de trabalho.

Essas tarefas são elaboradas seguindo os mesmos preceitos das listas de exercícios; muda-se apenas o instrumento, mas o caráter e objetivo das atividades permanecem os mesmos. Novamente, a prática pode acontecer individualmente ou em grupo, e vale o requisito de que o ritmo de trabalho de cada aluno e o acompanhamento correspondente é individual.

3.5. Exemplos de atividades por fase

Primeira fase

Exemplo de exploração do método de tentativa e erro.

> PF1. Utilizando quatro quatros e as operações de soma, subtração, multiplicação, divisão e raiz quadrada, escrever os números de 0 a 9. Exemplo: zero = 4+4-4-4.

Exemplo de interpretação de enunciado.

> PF2. Uma gaveta contém 1483 meias brancas e 4237 meias vermelhas. Quantas meias, no mínimo, devem ser retiradas da gaveta para que se garanta pelo menos um par da mesma cor?

Transição entre Primeira e Segunda fases

Exemplo de exploração do método de tentativa e erro e elaboração de estratégia elementar.

> PSF1. Você tem em suas mãos duas jarras, uma com capacidade para quatro (4) litros e outra menor, com capacidade para apenas três (3) litros. Essas jarras são disformes e não têm marcações intermediárias, de forma que se qualquer uma das jarras não estiver completamente cheia não será possível saber quanta água haverá dentro dela. Há uma torneira disponível para sua utilização, e um ralo que permite que a água seja jogada fora. Você pode manipular as jarras das seguintes formas:
>
> 1. Encher completamente uma jarra usando a torneira;
> 2. Passar água de uma jarra para a outra;
> 3. Esvaziar completamente uma jarra jogando a água que estiver dentro dela no ralo.
>
> O seu objetivo é colocar dois (2) litros exatos de água em qualquer uma das jarras.

Segunda fase

Exemplo de elaboração e formalização de estratégia.

> SF1. Um canoeiro deseja levar seus três pertences de um lado do rio para outro, porém a sua canoa só comporta, além do canoeiro, um pertence de cada vez. Os pertences em questão são uma cabra, um maço de couves e um lobo. Diga como o canoeiro pode fazer para poder transportar todos os seus pertences para o outro lado, levando em conta que:
>
> Se a cabra ficar com a couve sem o canoeiro por perto, ela come a couve;
> Se o lobo ficar com a cabra sem o canoeiro por perto, ele come a cabra;

Exemplo de elaboração e formalização de estratégia de nível elevado de abstração

> SF2. A "Torre de Hanói" é um famoso brinquedo que consiste de três (3) hastes verticais espaçadas e cinco (5) discos de tamanhos diferentes, furados no meio. Inicialmente os três discos encontram-se enfiados na primeira haste, o maior mais em baixo, o de tamanho médio no meio e o menor mais em cima. O objetivo deste jogo é mover os cinco discos para a terceira haste, levando em conta que não é permitido que um disco de tamanho maior seja colocado em cima de um menor, e que só se pode mover um disco por vez, da seguinte forma: retirar o disco que estiver mais em cima em uma das hastes verticais e colocá-lo no topo da pilha de discos que estiver enfiada em outra haste vertical. Explique como devemos mover os discos, de forma a atingir o resultado final desejado, sem violar as restrições do jogo.

Transição entre Segunda e Terceira fases

Exemplo de elaboração de estratégia interativa e formalização orientada a terceiros.

> STF1. Escreva uma estratégia para jogar o jogo da velha e nunca perder.

Terceira fase

Exemplo de elaboração de procedimento simples e formalização orientada a terceiros.

> TF1. Diga a seqüência de passos que você faz para vestir um traje completo composto de: calça, blusa, casaco, meias, e sapatos. Mas atenção: repare que você só pode colocar uma peça de cada vez.

Exemplo de identificação de processo repetitivo.

> TF2. Diga a seqüência de passos que você faz para descobrir a posição de um nome em uma lista de chamada ordenada alfabeticamente.

Exemplo de identificação de processo recursivo.

> TF3.Torre de Hanói com n discos

4. A Implantação

Nos cinco primeiros anos de existência do curso de ciência da computação das FIB, constatou-se deficiência considerável no processo de aprendizagem nas disciplinas da linha de programação, principalmente na disciplina introdutória de estudo de algoritmos. Esta discussão veio a tona durante a elaboração do novo projeto pedagógico e o resultado foi a decisão de mudar a abordagem, adotando uma primeira versão da metodologia aqui descrita. Ao longo do próprio período de implantação, segundo semestre de 2003, os resultados foram discutidos em encontros pedagógicos dos quais participaram, além dos professores, alunos e ex-alunos. A conseqüência desses encontros foi a evolução da metodologia para o formato apresentado neste trabalho.

A primeira versão da metodologia foi adotada a partir de 2003.2 na disciplina Introdução ao Estudo de Algoritmos do primeiro período. O programa da disciplina incluía considerável conteúdo de programação. Não havia aulas expositivas de qualquer espécie, apenas aulas práticas. O andamento era individual, sem atividades em grupo e com pouca interferência do professor. Nessa ocasião foram estabelecidas duas fases, que coincidiam com a primeira e a terceira do formato hoje adotado. A fase de

resolução de problemas foi bem desenvolvida. A tentativa de construir algoritmos, na fase seguinte, foi prejudicada pela deficiência na formalização. Esta habilidade foi trabalhada somente no semestre seguinte com a criação da fase de formalização.

As conseqüências positivas foram a redução quase total da evasão, embora mantido o nível de reprovação, e um aumento nos níveis de motivação. As avaliações dos procedimentos adotados apontaram que melhores índices poderiam ser alcançados com maior interação entre alunos e alunos-professor durante as aulas. Isso, além de ser fator de motivação, ajudaria a suavizar a defasagem nos estágios de aprendizagem dos alunos.

Nessa versão o foco da metodologia ainda estava no conteúdo de programação. A resolução de problemas era considerada um instrumento de potencialização. Esse foi outro fator identificado como restritivo, reforçado pela falta do trabalho de formalização, gerando um vácuo na relação entre a fase de resolução de problemas com o objetivo de construção de algoritmos.

A medida de maior impacto na correção do processo foi a redução de conteúdo de programação no semestre seguinte, aprovada no Colegiado do Curso. O foco foi transferido para o aluno e seu desenvolvimento. Foi criada a fase correspondente à formalização. Programaram-se aulas introdutórias e sincronizadoras bem como atividades em grupo, aumentando a interação. Os exercícios foram mais bem associados às fases e classificados segundo as habilidades que visam a desenvolver. Estas modificações deram origem à metodologia descrita.

5. Conclusão

A metodologia proposta está em seu primeiro período de adoção, mas já é uma evolução de uma tentativa anterior. A aceitação dos alunos é boa e os reflexos em seu desenvolvimento são promissores. Todos os alunos são atingidos, não há exclusão. Portanto, diminuem as taxas de evasão e o aluno cria uma expectativa de sucesso que se reflete em seu envolvimento. A participação mais interessada e a melhoria progressiva no nível de resultados nas atividades propostas são bons indicadores.

O acompanhamento dos alunos, através da observação dos seus desempenhos com as listas de exercício, traz indícios de uma curva de aprendizado mais suave. A mudança está no fato do andamento do curso estar centrado na evolução do aprendizado e não na exposição de unidades didáticas. O compromisso temporal com o conteúdo fragmenta a evolução e o aprendizado diminui a cada nova unidade didática porque o pré-requisito – o aprendizado da fase anterior - não se completou. O aluno sem vocação natural, com os conteúdos anteriores comprometidos, progride muito devagar e, ou tem um “estalo de Vieira”, ou continua na mesma. A metodologia proposta deve garantir um aprendizado contínuo porque não carrega bagagem.

As metodologias utilizadas nas disciplinas subseqüentes também precisam ser revistas para que o processo de aprendizagem do aluno mantenha suas características evolutivas.

Referências

Burton, P. J. e Bruhn, R. E. (2003) “Teaching Programming in the OOP Era”, SIGCSE Bulletin, 35(2): 111--114.

Carvalheiro, F. e Setzer, V. (1995) “Uma Introdução Geral aos Algoritmos”, Anais do 3º WEI, 4º Congreso Ibero Americano de Educación Superior em Computación, 15º Congresso da SBC 1995, Brasil.

Chaves de Castro, T., Castro Júnior, A., Menezes, C., Boeres, M. e Rauber, M. (2003) “Utilizando Programação Funcional em Disciplinas Introdutórias de Computação”, Anais do WEI 2003, Brasil.

Falkembach, G. (2003) “Uma experiência de Resolução de Problemas através da Estratégia Ascendente”, I workshop do grupo de pesquisa em cognição e computação, ULBRA Canoas - RS, Brasil.

Giraffa, L. (2003) “O ensino de algoritmos e programação mediado por um ambiente na Web”, Anais do WIE 2003, Brasil.

Kortright, L. (1994) “Technical Symposium on Computer Science Education”, Proceedings of the twenty-fifth SIGCSE symposium on Computer science education, ACM Press , EUA, pp 71-75.

Levitin, A. (2000) “Design and analysis of algorithms reconsidered”, Technical Symposium on Computer Science Education, Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, EUA, pp 16-20.

Magri, J., Lógica de Programação - Ensino Prático, Érica, 2003.

Marion, W. (1999) “CS1: what should we be teaching?”, Annual Joint Conference Integrating Technology into Computer Science Education ,Working group reports from ITiCSE on Innovation and technology in computer science education, pp 35-38, Polônia.

Mckeown, J. e Farrell, T. (1999) “Why We Need to Develop Succcess in Introductory Programming Courses”, CCSC – Central Plains Conference, Maryville, MO.

Padro, B. e Maria Elisabette B., “(Re)visitando o construcionismo para a formação do professor reflexivo”, disponível em URL <http:// www.c5.cl/ieinvestiga/ actas/ ribie98/ 239.html em 18/04/2004.

Perego, C., Lisboa, M. e Bertagnolli S. (2002) “A Migração de Pascal para Java: Problemas e Propostas de Solução”, Anais do WIE 2002, Brasil.

Polya, G., How to solve it, Princeton Univ. Press, 1957.

Rosso, A. e Daniela, M. (2000) “Our method to teach algorithmic development”, ACM SIGCSE Bulletin, Volume 32, Issue 2, ACM Press, EUA, pp 49-52.

Santos, R., Xexéo, J., Rapkiewicz, C., Campos, M. e Machado, M. (2003) “Inovando a dimensão estratégica do ensino de Ciência da Computação”, WEI 2003, Brasil.

Souza, J., Lógica para Ciência da Computação, Campus, 2002.

Suraweera, F. (2001) “Getting the most from an algorithms design course: a personal experience”, Volume 33 , Issue 4, ACM Press, EUA, pp 71-74.