Ensino de Algoritmos baseado na Aprendizagem Significativa utilizando o Ambiente de Avaliação NetEdu

Edson Pinheiro Pimentel [1], Nizam Omar [2]

[1] Universidade Municipal de São Caetano do Sul (IMES)
Av. Goiás, 3400 – 09550-051 – São Caetano do Sul– SP – Brasil

[2] Universidade Presbiteriana Mackenzie
Rua da Consolação, 930 – 01302-907 – São Paulo – SP – Brasil

edson.pimentel@imes.edu.br, Omar@mackenzie.br

Abstract. *Learning a Computer Programming Language is a difficult and demanding process. This is shown by the high disapproval rate in disciplines of Introductory Algorithms and also in advanced disciplines of this field. The way of knowledge organization happens in instructional processes can be crucial for learning outcomes. This article describes a methodology to identify gaps in the teaching and learning of algorithms. By using a prototype of an assessment environment it is possible to set up the relations of pre-requisites between the whole content and to create connections in each unit of assessment. As result, the environment provides information for monitoring the performance of students in a specific content.*

Resumo. *Aprender a programar é um processo difícil e exigente. Isto pode ser constatado pelo alto índice de reprovação em disciplinas introdutórias de Ensino de Algoritmos e também em disciplinas avançadas. A forma como a organização do conhecimento ocorre em processos instrucionais pode ser determinante nos resultados da aprendizagem. Este artigo descreve uma metodologia para identificação de lacunas de aprendizagem aplicada no Ensino de Algoritmos. Com o uso de um protótipo de ambiente de avaliação estabelecem-se relações de pré-requisitos entre os conteúdos e criam-se conexões desses com cada unidade de avaliação. Como resultado, o ambiente permite acompanhar os resultados do estudante em cada conteúdo.*

1. Introdução

As problemáticas e dificuldades pertinentes ao aprendizado de algoritmos constituem um desafio para os estudantes e docentes. As dificuldades encontradas podem ser diagnosticadas não somente pelo alto grau de repetência, mas também pelos problemas demonstrados por alunos em disciplinas avançadas que exigem o pré-requisito de algoritmos [Rocha, 1991; Gomes e Mendes, 2000].

Aprender a programar constitui-se de uma tarefa complexa que inclui: adquirir habilidade para resolução de problemas, aprender a sintaxe e a semântica de uma linguagem de programação, utilizar um ambiente de programação e realizar testes e depuração de programas [du Bolay, 1986]. É preciso identificar detalhadamente em quais dessas tarefas existem as lacunas que impedem ou dificultam o aprendizado.

As lacunas de aprendizagem causadas por ausência de pré-requisitos bem definidos contribuem para ampliar as dificuldades de aprendizagem e levam ao insucesso escolar de muitos alunos. Uma metodologia de ensino-aprendizagem deveria dar condições ao aluno e professor para identificação destas lacunas e num segundo momento se preocupar em suprir o que falta aprender, ou seja, instituir um ensino pautado nas necessidades individuais de aprendizagem. A identificação do que se sabe e do que não se sabe pode ser obtido através de avaliações contínuas, integradas ao processo de aprendizagem e não apenas em momentos finais [Masetto, 2003].

A forma como a organização do conhecimento ocorre em processos instrucionais pode ser determinante nos resultados da aprendizagem. Os conteúdos listados nas ementas das disciplinas servem de guia para que os professores conduzam suas aulas e estes são predominantemente organizados de maneira seqüencial, mas sem especificar as relações entre os conteúdos. Isto impede que se possa estruturar um caminho para a recuperação do estudante em caso de insucesso, pois o mesmo não consegue identificar claramente os pré-requisitos que não possui.

Este trabalho descreve uma metodologia para identificação de lacunas de aprendizagem aplicada no Ensino de Algoritmos. Para experimentar a metodologia proposta construiu-se um protótipo de ambiente de avaliação que permite estabelecer relações de pré-requisitos entre os conteúdos e criar conexões desses com cada unidade de avaliação. Ao fazer a correção da unidade de avaliação, verificando acertos e erros, o professor anota o percentual de acerto em cada item avaliado e como resultado o ambiente permite acompanhar a evolução do desempenho do estudante em cada conteúdo, em várias avaliações.

O artigo está organizado como segue. A seção 2 discute sobre o Ensino de Algoritmos e a seção 3 descreve a Aprendizagem Significativa e a Teoria dos Espaços do Conhecimento. Na seção 4, apresenta-se a Organização do Conhecimento com Mapas Conceituais. Na seção 5 detalha-se o ambiente NetEdu e o seu uso no Ensino de Algoritmos. Finalmente, na seção 6 são feitas algumas considerações acerca desse trabalho e os aprofundamentos necessários.

2. Ensino de Algoritmos

O ensino das linguagens de programação de computadores tem por objetivo capacitar os alunos a desenvolver programas e sistemas computadorizados que objetivam a resolução de problemas. Entretanto aprender a programar não é fácil e isto se reflete nos baixos níveis de assimilação experimentados por muitos alunos nos cursos de programação introdutórios [du Boulay, 1986].

Os problemas enfrentados pelos alunos no aprendizado de programação têm preocupado os educadores da área. Na prática, muitos alunos ainda acham que o processo de aprendizagem de programação é difícil [Garner, 2000]. Mendes [2001] cita algumas causas destas dificuldades, entre as quais destacam-se: o elevado nível de abstração envolvido; as metodologias tradicionais de ensino, que privilegiam a aprendizagem de conceitos dinâmicos utilizando principalmente abordagens e materiais de natureza estática; existência de turmas com backgrounds diversificados acerca das matérias em questão, traduzindo-se em ritmos de aprendizagem muito diferenciados; a impossibilidade de um acompanhamento individualizado do aluno, etc. Estes fatores

podem levar o aluno de linguagem de programação, principalmente os iniciantes, a sentir necessidade de um acompanhamento e orientação que o professor nem sempre disponibiliza e que os métodos de ensino tradicionais nem sempre dão resposta.

Ferramentas como o C-Tutor [Song et al., 1997] e SICAS [Gomes e Mendes, 2000] foram desenvolvidas com o propósito de facilitar o aprendizado de lógica e linguagens de programação, porém há ainda uma carência de metodologias nestes ambientes que possibilitem tratar cada aprendiz de maneira diferenciada. Ou seja, as metodologias geralmente são aplicadas de maneira uniforme em turmas inteiras e como destaca Cardoso e Jandl [1998] alunos não são iguais porque possuem origens, experiências e habilidades diferentes uns dos outros, o que reforça a necessidade do uso de técnicas variadas que permitam ampliar os resultados de ensino. Pesquisas conduzidas por Anderson e Skwareck [1996] revelaram que Sistemas Tutores Inteligentes (STI) são ferramentas eficazes neste domínio, pois fornecem atenção individualizada a cada aprendiz para superar as dificuldades de aprendizagem.

3. Aprendizagem Significativa e a Teoria dos Espaços do Conhecimento

A idéia central de David Ausubel [Moreira e Masini, 2001] é a de que o fator isolado mais importante influenciando a aprendizagem é aquilo que o aprendiz já sabe. Para ele, aprendizagem significativa é um processo pelo qual uma nova informação se relaciona com um aspecto relevante da estrutura de conhecimento do indivíduo, ou seja, a aprendizagem ocorre quando a nova informação ancora-se em conceitos relevantes preexistentes na estrutura cognitiva de quem aprende.

Na aprendizagem significativa as novas idéias e informações interagem com um conhecimento prévio existente na estrutura cognitiva do indivíduo, definido por Ausubel como sendo idéias-âncora (subsunçores). Trata-se de uma idéia (conceito) mais ampla, que funciona como subordinador de outros conceitos na estrutura cognitiva e como "âncora" no processo de assimilação. Como resultado dessa interação (ancoragem), a própria idéia-âncora é modificada e diferenciada.

Portanto, para Ausubel, a variável crucial para a aprendizagem significativa é a estrutura cognitiva do indivíduo, ou seja, os seus conhecimentos prévios, que deverão funcionar como idéias-âncora à assimilação de novos conhecimentos. A assimilação ocorre quando um novo significado **“a”**, adquirido em ligação com idéias-âncora **“A”** com as quais está relacionado, é retido e ocorre uma modificação decorrente da intersecção de ambos, **A'a'**.

Para dar suporte ao emprego da aprendizagem significativa no Ensino de Algoritmos esse trabalho baseou-se na Teoria dos Espaços do Conhecimento desenvolvida por Doignon e Falmagne (1998) e criou uma estrutura na forma de hierarquia de conceitos organizados através de pré-requisitos de aprendizagem.

Para Doignon e Falmagne (1998) se Q é um conjunto de itens, o estado de conhecimento de um estudante pode ser descrito como o subconjunto de Q. Em virtude das relações de pré-requisitos entre os itens, o conjunto de possíveis estados de conhecimento, o *espaço do conhecimento*, é restrito ao domínio de valores de Q. Uma maneira de representar trais relações de pré-requisitos é a “relação presumida”.

Dois itens, x, y ε *Q* estão em relação de pré-requisito se, de uma resposta correta para o item y, pudermos assumir uma resposta correta para o item x. Cada relação presumida descrê um único espaço de conhecimento. Relações presumidas podem ser representadas através de diagramas de Hasse conforme ilustrada na figura 1 extraída de Hockemeyer e Albert (1999). Os círculos representam conceitos e os quadrados representam estados do conhecimento. Assim na figura 1b, o quadrado no topo indica que o estudante possui todo o conhecimento do conjunto Q = { w,x, y, z}.

Aplicando-se a teoria de espaços do conhecimento em ambientes de aprendizagem, pode-se obter o conceito de "caminhos de aprendizagem" descrevendo possíveis trajetos que um estudante pode seguir num curso específico.

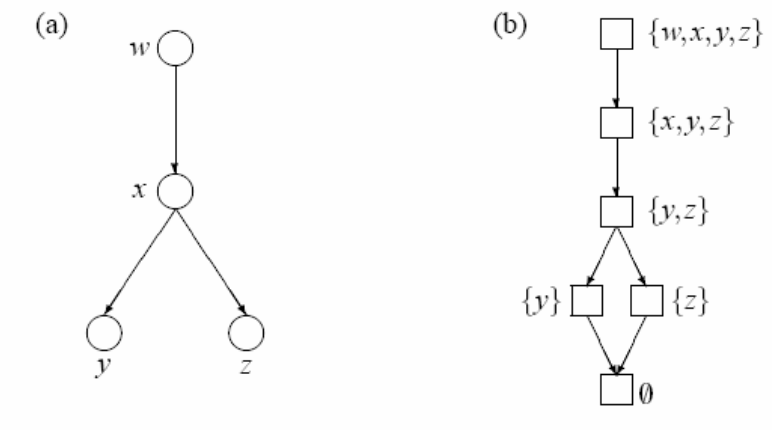

Figura 1 – Relação Presumida (a) e Estados do Conhecimento (b) para problemas em Q.

A figura 2 apresenta um exemplo de espaços de conhecimento aplicado a uma parte importante do conteúdo de "Lógica de Programação". Os números ao lado de cada nó (conceito) indicam o grau de entrada de cada conteúdo, ou seja, o grau de dependência. Assim, o nó "5" (Condição) possui grau de entrada "3" pois depende direta ou indiretamente dos conceitos "1", "2" e "3".

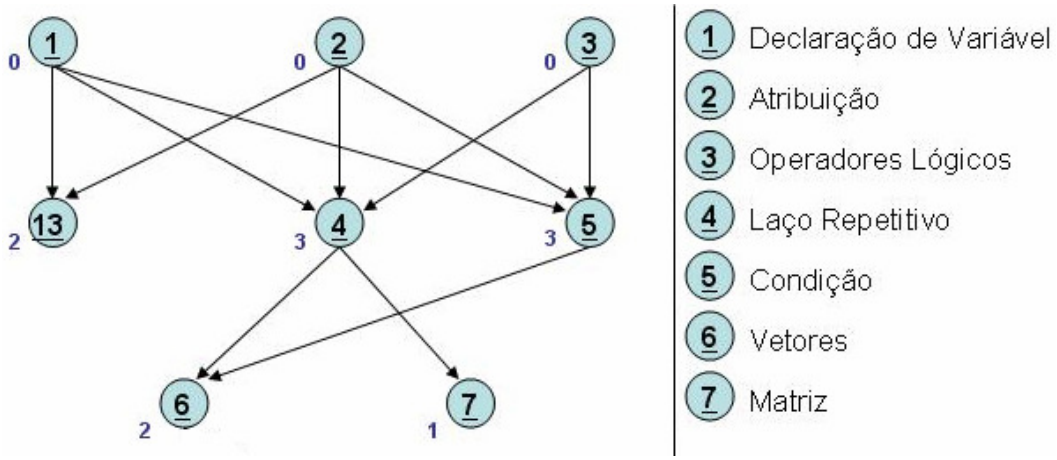

Figura 2 – Exemplo de Hierarquia de Conceitos num Espaço de Conhecimento de Lógica de Programação

4. Organização do Conhecimento com Mapas Conceituais

Considerando as recomendações da aprendizagem significativa de que conceitos mais gerais devem ser apresentados inicialmente como organizadores prévios, o código apresentado na figura 2 poderia ser utilizado numa aula introdutória para mostrar os componentes de "programa de computador" Isso com certeza surtiria mais efeito do que apenas apresentar o conceito de "programa".

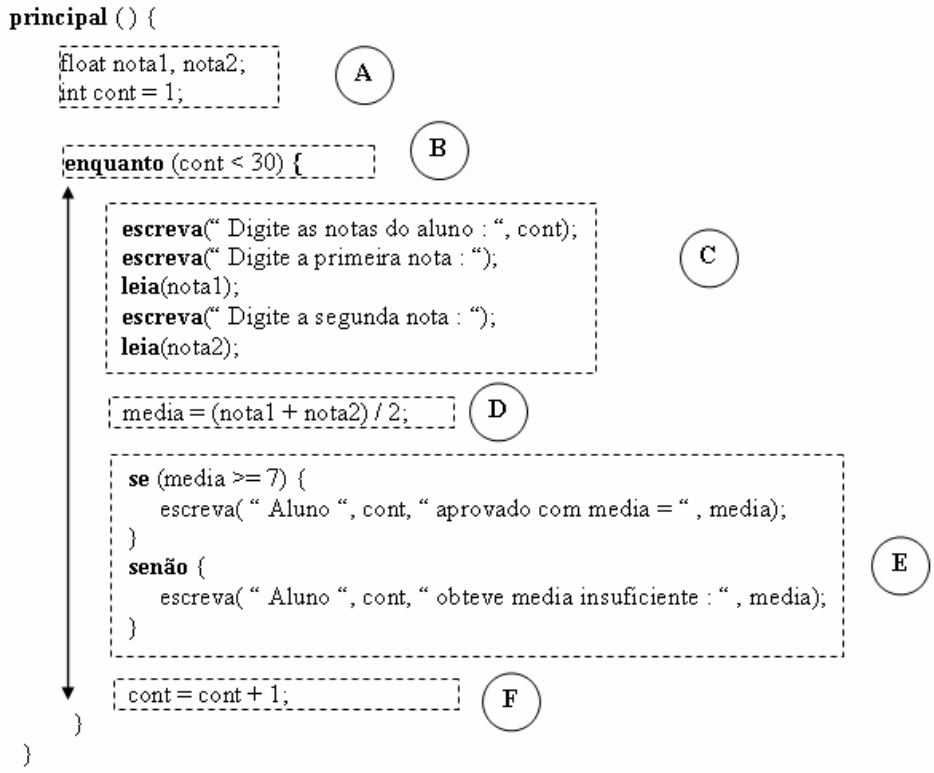

Figura 2. Um programa de computador: organizador prévio.

As letras de "A" a "F" apresentados ao lado de cada bloco do programa especificam blocos de conteúdos a serem ministrados no decorrer da disciplina. "Ensino de Algoritmos" conforme tabela 1.

Tabela 1. Conceitos ministrados numa disciplina de Lógica de Programação.

Código	Descrição do Conceito
A	Declaração de Variável
B	Estruturas de Repetição
C	Entrada de Dados
D	Expressões Aritméticas
E	Estruturas de Seleção
F	Acumulação

No entanto, apenas listar conteúdos numa seqüência ou mesmo agrupa-los em blocos (tópicos gerais) não permite visualizar a dependência de conceitos. A figura 3 apresenta um Mapa Conceitual utilizando uma gama de conceitos utilizados no Ensino de Algoritmos, englobando aqueles que constam na tabela 1 e todas as relações de pré-requisitos.

Mapas Conceituais consistem de uma técnica desenvolvida por Joseph Novak (1998) para dar suporte a aprendizagem significativa através de um diagrama que indica relações entre conceitos (Moreira e Masini, 2001).

A próxima seção mostra como o uso da Teoria dos Espaços do Conhecimento e as relações entre conceitos construídas com o suporte de Mapas Conceituais implementadas num ambiente de avaliação formativa da aprendizagem podem dar suporte a aprendizagem significativa.

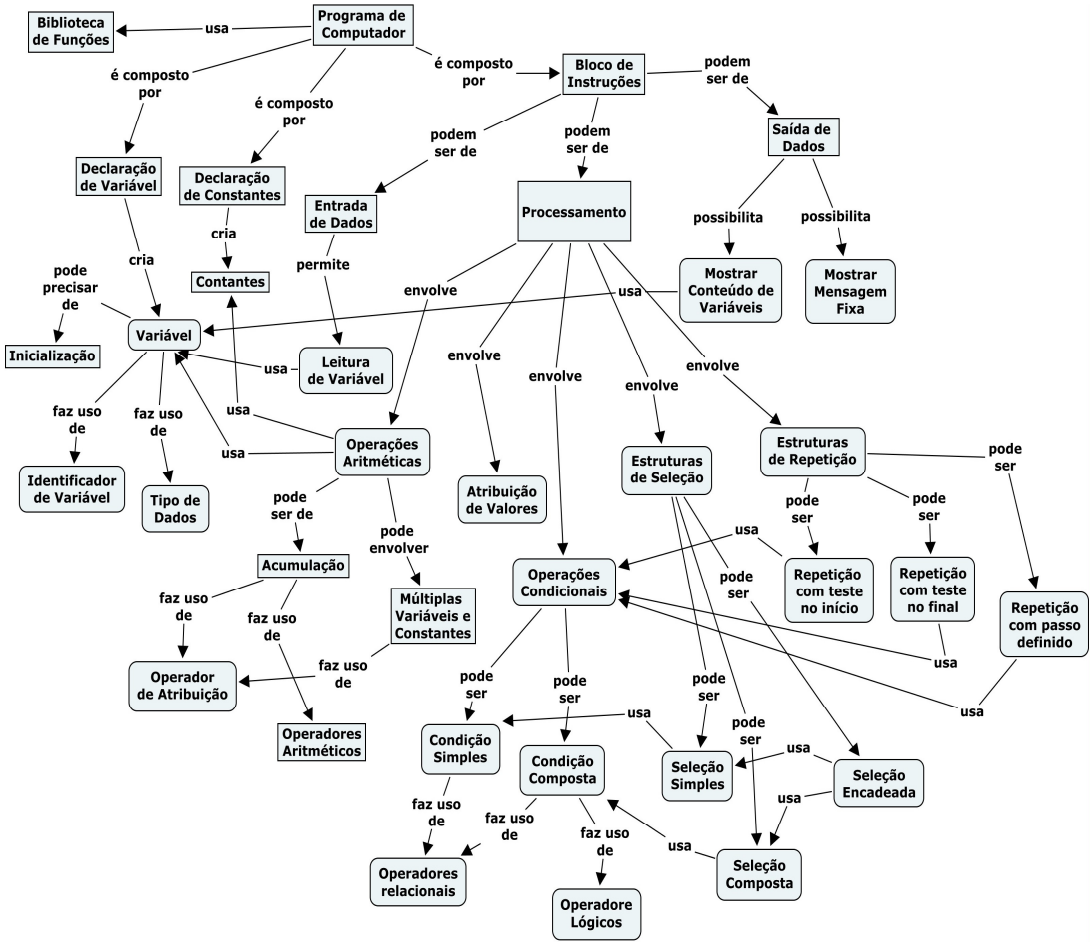

Figura 3. Mapa Conceitual - Conceitos abordados em Lógica de Programação.

4. NetEdu: Um ambiente de Avaliação Formativa

O NetEdu é um ambiente de avaliação formativa que baseia-se na arquitetura pedagógica proposta por **Omitido (yyyy)** e que tem como objetivo principal auxiliar o professor e estudante a identificarem o Nível de Aquisição de Conhecimentos (NAC) do estudante.

Segundo **Omitido (xxxx)** NAC é uma medida instantânea que indica o grau de conhecimentos do aprendiz num determinado conteúdo de um domínio de conhecimento e pode ser obtido a partir de diversas atividades de avaliação. Em cada avaliação deve-se atualizar o NAC nos conceitos envolvidos naquela avaliação. Têm-se, portanto a medida de desempenho naquela avaliação e também a medida histórica.

A figura 3 apresenta a arquitetura geral do NetEdu com as suas principais funcionalidades e seus quatro possíveis atores: administrador, professor, estudante e monitor.

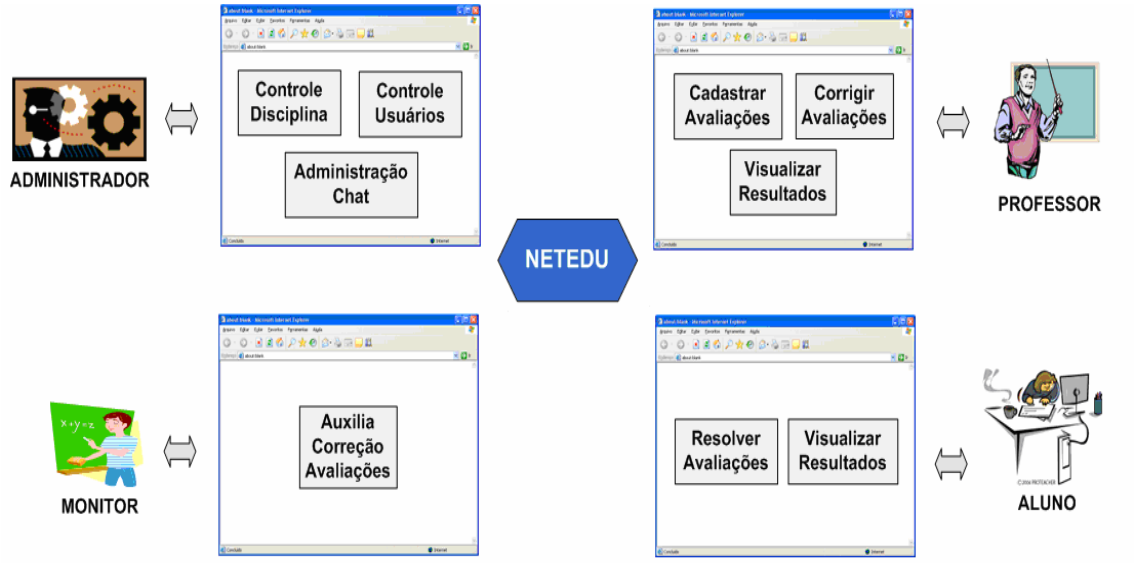

Figura 3. Principais funcionalidades do NETEDU

Após cadastrar previamente os conteúdos e as relações de pré-requisitos o professor pode criar unidades de avaliação (UA) associando a elas os conceitos avaliados e o peso de cada conceito naquela UA, conforme figura 4. O estudante, por sua vez, responde a estas questões e como resultado o NAC do estudante é atualizado em cada conceito avaliado naquela UA

Figura 4. Cadastro de Unidade de Avaliação e Associação de Conteúdos

Após a resolução do exercício por parte do estudante o professor ou um auxiliar (monitor) deverão atribuir o grau de acerto para a solução do estudante em cada conteúdo conforme Figura 7.

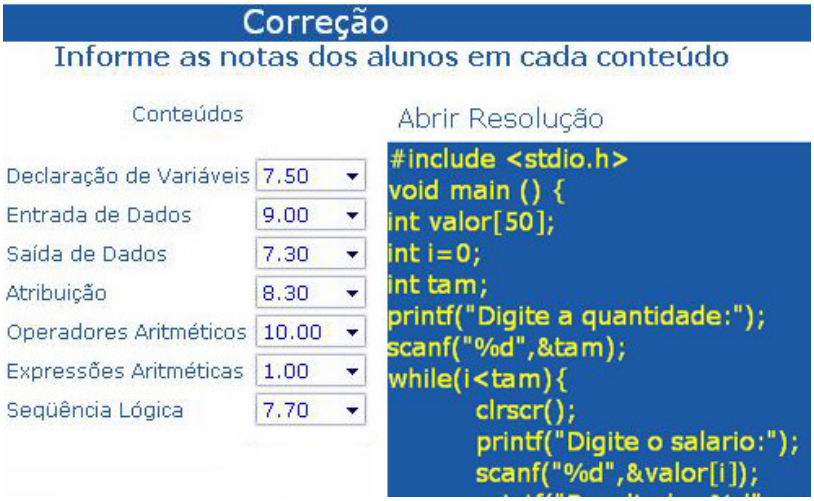

Figura 5. Atribuição de Grau de Acerto

A Figura 12 apresenta a tela de visualização do desempenho cognitivo no NetEdu. O estudante pode visualizar apenas os seus próprios resultados enquanto o professor pode ter acesso aos resultados de um único estudante ou de todo grupo.

Visualização de Resultados

Disciplina: Lógica de Programação

CONTEÚDOS	QUANTIDADE UA	NAC	AP	EMO	EMP	EGO	EGP	KMA	KMB
Declaração de Variáveis	1	0.0	0	0	0	1	0	-1.00	1.00
Entrada de Dados	0	0.0	0	0	0	0	0	0.00	0.00
Saída de Dados	1	0.3	0	1	0	0	0	-0.50	0.50
Atribuição	0	0.0	0	0	0	0	0	0.00	0.00
Operadores Aritméticos	0	0.0	0	0	0	0	0	0.00	0.00
Expressões Aritméticas	1	0.3	1	0	0	0	0	1.00	0.00
Seqüência Lógica	1	0.6	0	1	0	0	0	-0.50	0.50
Operadores Relacionais	0	0.0	0	0	0	0	0	0.00	0.00
Expressões Lógicas	0	0.0	0	0	0	0	0	0.00	0.00
Seleção	0	0.0	0	0	0	0	0	0.00	0.00
Repetição	1	1.1	0	0	1	0	0	-0.50	-0.50
Declaração de Constantes	1	0.0	1	0	0	0	0	1.00	0.00
Declaração de Vetor	1	0.6	0	1	0	0	0	-0.50	0.50
Acesso ao Vetor	1	0.6	1	0	0	0	0	1.00	0.00

Figura 6 - Resultados Históricos de Desempenho

A tabela 2 apresenta um exemplo de como esta metodologia permite o acompanhamento da aprendizagem do aprendiz num processo de avaliação contínua, além de indicar pontualmente as lacunas de aprendizagem. A coluna "GC" indica os valores normalizados do grau de confiança do aprendiz, em cada item, obtido com um questionário inicial. As colunas NAC-1 e NAC-2 apresentam o Nível de Aquisição de Conhecimentos do aprendiz em cada item, respectivamente após a resolução dos problemas P1 e P2 (NAC-1) e P3 e P6 (NAC-2)

Tabela 2. Acompanhamento da Aprendizagem de um Aluno

Conteúdo		GC	P1	P2	NAC-1	P3	P6	NAC-2
1.1.1	Declaração de Variáveis	**1,00**	1,00	1,00	**1,00**	0,50	0,40	**0,73**
2.1	Atribuição	**0,60**	1,00	1,00	**1,00**	1,00	0,00	**0,75**
2.2	Operadores Aritméticos	**1,00**	1,00	1,00	**1,00**	0,70	0,00	**0,68**
2.3	Operadores Relacionais	**0,40**	-	1,00	**1,00**	0,70	0,70	**0,80**
3	Expressões Aritméticas	**0,60**	0,80	1,00	**0,90**	0,70	0,00	**0,63**
5.1	Entrada de Dados	**0,80**	0,30	0,60	**0,45**	0,50	0,40	**0,45**
6.1	Saída de Dados	**0,60**	1,00	1,00	**1,00**	1,00	1,00	**1,00**
7	Estruturas de Seleção	**0,60**	-	0,80	**0,80**	0,80	0,70	**0,77**
8	Estruturas de Repetição	**0,60**	-	-	**-**		0,00	**0,00**

Observa-se, por exemplo, que para o item 7 – Estruturas de Seleção, este aprendiz parte de um GC = 0,60 que poderia ser o NAC inicial e evolui após a resolução do problema P1 e P2 para um NAC-1 = 0,80. No entanto, após a resolução dos problemas P3 e P6, o NAC-2 cai para 0,77 indicando que o NAC do aprendiz pode aumentar ou diminuir a cada instante. No item 8 – Estruturas de Repetição, este aprendiz parte de um GC = 0,60 e chega a um NAC-2 = 0,00 indicando que esta pode ser uma Lacuna de Aprendizagem para o mesmo. Esta oscilação no NAC do aprendiz, em cada item pode ser melhor visualizado através do gráfico da Figura 7.

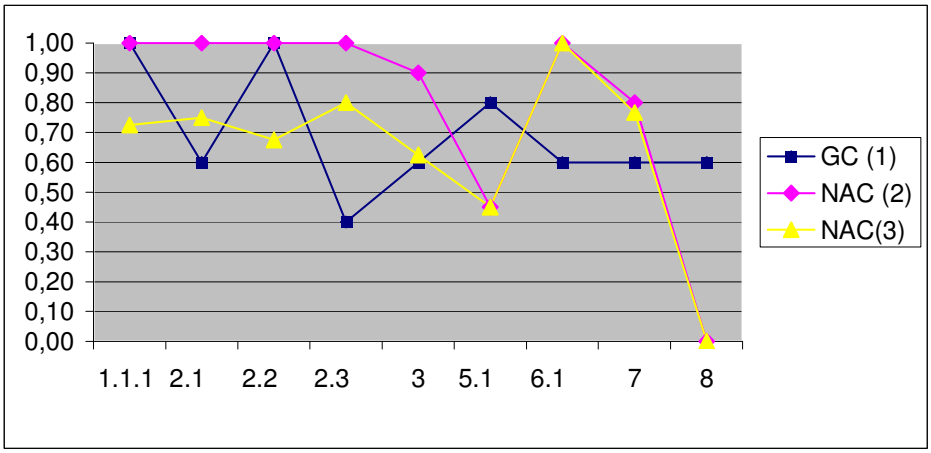

Figura 7 – Evolução do NAC do Aprendiz

7. Conclusões e Aprofundamentos Necessários

As dificuldades encontradas no Ensino de Algoritmos visível na dificuldade de aprendizagem dos alunos e nos altos índices de reprovação, tem demonstrado ser um terreno fértil para pesquisas que busquem a identificação de novas metodologias, principalmente apoiadas em ambientes computacionais.

Este trabalho apresentou uma metodologia apoiada na Aprendizagem Significativa com o objetivo de identificar pontualmente lacunas de aprendizagem. Para tal fez-se necessária a organização do conhecimento numa estrutura de pré-requisitos. A proposta foi experimentada no ambiente de avaliação NetEdu implementado com esse

propósito. Verificou-se que o mesmo permite um acompanhamento mais preciso do Nível de Aquisição do Conhecimentos do estudante em cada conteúdo.

Como pontos que merecem aprofundamentos identificou-se a necessidade de se pesquisar a correção automática de algoritmos para agilizar o feedback para o estudante e também realizar estudos de impactos sobre a aprendizagem a partir do mapeamento das lacunas de aprendizagem do estudante.

Referências

Anderson J.R. e Skwareck E. (1996). The Automated Tutoring of Introductory Computer Programming, Communications of the ACM, Vol. 29, pp. 842-849, ACM Press.

Cardoso, S. M. V. e Jandl, Peter (1998). *Estilos de Aprendizagem: Aprender a Aprender*.

Doignon, J.-P. & Falmagne, J.-C. (1998). Knowledge Spaces. Berlin: Springer–Verlag.

du Boulay, J. B. H. (1986) Some Difficulties of learning to program. In: Journal of Educational Computing Research, 2(1):57-73.

Hockemeyer, C., & Albert, D. (1999). The Adaptive Tutoring System RATH. In M. E. Auer & U. Ressler (Eds.), ICL99 Workshop Interactive Computer aided Learning: Tools and Applications. Villach, Austria: Carinthia Tech Institute.

Garner, S.K.(2000) A Code Restructuring Tool to help Scaffold Novice Programmers, International Conference in Computer Education ICCE2000, Taipei, Taiwan.

Gomes, A. e Mendes, A. (2000). *Suporte à aprendizagem da programação com o ambiente SICAS*. Actas do V Congresso Ibero-Americano de Informática Educativa, Viña del Mar.

Maia, Rodrigo F. (2004) Sistema Multi-Agentes para Acompanhamento e Auxílio de Avaliação de Alunos em Ambientes de Ensino à Distância, Dissertação de Mestrado, Escola Politécnica da Universidade de São Paulo.

Masetto, Marcos T. (2003). Competência Pedagógica do Professor Universitário. São Paulo: Summus.

Moreira, Marco A., Masini, Elcie F.S. (2001). Aprendizagem Significativa: a teoria de David Ausubel. São Paulo: Centauro.

Perrenoud, Philippe. (1999) Avaliação: da excelência à regulação das aprendizagens – entre duas lógicas. Porto Alegre: Artmed Editora.

Rocha, Helena. V. (1991). Representações Computacionais Auxiliares ao Entendimento de Conceitos de Programação. Unicamp.

Song, J. S., Hahn, S. H., Tak, K. Y. e Kim, J. H. (1997) *An Intelligent Tutoring System for Introductory C Language Course*. Computers Education. Vol 28 No. 2.