Avaliação Contínua da Aprendizagem, das Competências e Habilidades em Programação de Computadores

Edson P. Pimentel[1,2], Vilma F. de França[1], Robinson V. Noronha[1], Nizam Omar[1,3]

[1]Instituto Tecnológico da Aeronáutica – (ITA)
Praça Mal. Eduardo Gomes, 50 – 12228-900 – São José dos Campos – SP – Brasil

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

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

epiment@imes.edu.br, vilmaff@comp.ita.br,
rvida@dainf.cefetpr.br, omar@mackenzie.com.br

Abstract. *This paper describes a learning assessment and continuous accompaniment model to qualify students in Computer Programming domain. It's intended with the model to create conditions of a more accurate evaluation students' mental state, as well as supplying teachers and coordinators distinguished information of each learner in the related knowledge domain, allowing to take decisions to rehabilitate students with located deficiencies.*

Resumo. *Este artigo descreve um modelo de avaliação e acompanhamento contínuo da aprendizagem para a capacitação do estudante no domínio Programação de Computadores. Pretende-se com este modelo criar condições de uma avaliação mais exata do estado mental do estudante, bem como dotar professores e coordenadores de informações particularizadas de cada aprendiz no referido domínio de conhecimento, o que possibilitará a todos tomar decisões para reabilitar estudantes com deficiências localizadas.*

1. Introdução

Ensino e aprendizagem são dois aspectos de um fenômeno conhecido por ensino-aprendizagem. A natureza do ensinar tem como compromisso assegurar que todos aprendam. Segundo David Ausubel, o pai da Aprendizagem Significativa, o fator mais importante influenciando a aprendizagem é aquilo que o aprendiz já sabe [Moreira 2001]. É necessário, então determinar continuamente o que o aluno conhece e ensiná-lo de acordo. Para tal, faz-se necessário a definição de um modelo e de ferramentas que possam auxiliar os professores e a instituição de ensino nesta tarefa. Este trabalho apresenta aspectos e funcionalidades de um modelo de acompanhamento da aprendizagem apoiado por um STI, aplicado ao domínio de Programação de Computadores em cursos de graduação em Informática.

A aprendizagem de programação de computadores é essencial para todas carreiras ligadas a Informática. Programação é, sem dúvida a disciplina mais importante para a formação daqueles que terão no desenvolvimento de softwares o produto final do

seu trabalho. Uma vez que a aprendizagem de programação ocorre praticamente durante todo o curso, o baixo índice de assimilação dos estudantes nas disciplinas cujos requisitos exigem o conhecimento de programação tem sido um grande problema enfrentado em muitas instituições. [Rocha 1991] afirma que estamos tendo um fracasso no ensino de programação e [Gomes e Mendes 2000] fala do insucesso generalizado verificado na aprendizagem de programação.

O artigo está organizado, como segue. A seção 2 apresenta elementos sobre estilos de aprendizagem de programação. Na seção 3 discute-se o sistema de avaliação existente nas disciplinas de Programação de Computadores. Na seção 4 é apresentado o modelo proposto para avaliação e acompanhamento contínuo da aprendizagem. Na seção 5, são apresentados dados iniciais da aplicação da metodologia. Na seção 6 são feitas algumas considerações acerca deste trabalho e resultados esperados.

2. Aprendizagem de Programação

O ensino das linguagens de programação tem por objetivo capacitar os alunos a desenvolverem programas e sistemas computadorizados capazes de resolver problemas do mundo real. Aprender a programar é um processo difícil e exigente para a maioria dos alunos. A prática de ensino tem demonstrado que uma parcela significativa dos alunos de disciplinas introdutórias de programação apresenta grande dificuldade em compreender e aplicar certos conceitos abstratos de programação.

Diversos tipos de ferramentas e ambientes têm sido propostos com o objetivo de facilitar o aprendizado de lógica e linguagens de programação. Dentre outras, podemos destacar: ASTRAL [Rezende e Garcia 1995], AUTOMATA SIMULATOR e IC [Jandl e Zuchini 1999], C-Tutor [Song et al. 1997], BALSA-II [Brown 1988], ZEUS [Brown 1991], SICAS [Gomes e Mendes 2000], etc. No entanto, não existem indicações de uma vasta utilização destas ferramentas, que acabam por se tornar apenas resultados de produtos de investigação, sem grande divulgação junto à maioria dos professores e alunos.

Apesar de várias metodologias propostas terem verificado melhores índices de aprendizado no domínio de Programação, não se encontrou na literatura pesquisada, metodologias que possibilitem tratar cada aprendiz de maneira diferenciada. Alunos não são iguais: possuem origens, experiências e habilidades diferentes. Isto explica, em parte, o fato de alunos de uma mesma classe, submetidos às mesmas condições de ensino, apresentarem resultados distintos [Cardoso e Jandl 1998] e reforça a necessidade do uso de técnicas variadas que permitam ampliar os resultados de ensino.

No aprendizado de programação, além de conceitos básicos existem conceitos difíceis de serem assimilados, como recursão e passagem de parâmetros [Rocha 1991]. Os principais erros apresentados pelos aprendizes durante o treinamento de perícia de programação são: erros de sintaxe e semântica, dificuldades na compreensão do enunciado dos problemas e na concepção de algoritmos, incapacidade de detectar erros de lógica de programação, etc [Gomes e Mendes 2000]. Todos estes fatores levam o aluno de Programação, principalmente de disciplinas introdutórias, a sentir a necessidade de um acompanhamento personalizado que o professor nem sempre disponibiliza e que os métodos de ensino tradicionais nem sempre dão resposta, pois não se consegue identificar o problema localizado de cada aprendiz. Isto se deve principalmente à sistemática atual de avaliação discutida a seguir.

3. Avaliação da Aprendizagem de Programação: A Sistemática Atual

As dificuldades apresentadas na seção anterior podem ser diagnosticadas não somente pelo alto grau de repetência nas disciplinas introdutórias, mas também pelas dificuldades demonstradas pelos estudantes nas disciplinas avançadas que exigem o pré-requisito de programação. Com isto, muitos estudantes completam seus cursos sem as habilidades de programação desejadas. Em parte, isto é decorrência da dificuldade encontrada pelos professores para acompanharem efetivamente as atividades laboratoriais de programação, dado o grande número de estudantes geralmente sob sua supervisão [Tobar et al. 2001].

Os critérios de avaliação utilizados pelas instituições não são satisfatórios por dois motivos. Primeiro porque o fato de um aluno obter média 5.0 numa disciplina não indica que este aluno sabe 50% de tudo aquilo que foi abordado. Pode ser que ele tenha aprendido muito bem um tópico e quase nada de outros tópicos correlacionados. Com a ausência do aprendizado destes tópicos, o estudante acabará tendo problemas de acompanhamento nas disciplinas avançadas. Segundo, porque há pouca ou nenhuma comunicação entre os professores das várias disciplinas, sejam elas do mesmo semestre ou de semestres seguintes.

Observando-se o plano de desenvolvimento de disciplinas de programação, em diferentes instituições, constata-se que os critérios de avaliação não variam muito. Normalmente esta avaliação engloba uma prova e a elaboração de um projeto a cada bimestre, além da entrega de listas de exercícios [Kurnia et al. 2001]. Seguindo a sistemática de ensino de que "matéria dada é matéria aprendida e, portanto matéria cobrada", as provas acabam sendo a principal fonte para o diagnóstico do estado mental de cada aluno na disciplina. Se no cômputo geral o aluno conseguir atingir a média para aprovação ele prossegue no curso, caso contrário ele terá que refazer a disciplina, provavelmente junto com outras disciplinas de programação, pois poucas instituições trabalham com o pré-requisito formal, que impede o aluno de cursar uma disciplina mais avançada sem ter sido aprovado numa disciplina básica.

Vale ressaltar que não se defende aqui o uso de retenção como solução para o problema ensino/aprendizagem. Não se pode condenar o aluno a levar o problema de aprendizagem indefinidamente. Isto acaba acontecendo, pois todos os alunos são tratados de maneira uniforme, ou seja, a mesma aula é dada para quem já sabe muito, pouco ou nada sobre o assunto. Este projeto concentra-se na idéia de que é extremamente necessário definir um modelo que possibilite fazer um acompanhamento individualizado sobre o estado mental do estudante, em cada item dos cursos de programação, das disciplinas básicas até aquelas consideradas avançadas, para que os esforços sejam concentrados nas reais deficiências de cada estudante.

Com as ferramentas de avaliação empregadas pela prática de ensino, apenas o próprio aluno conseguiria identificar o seu estado mental em cada item, se ele fosse capaz de perceber efetivamente o que ele não sabe. No entanto os alunos não estão preparados para isto. Em conjunto com o acompanhamento da aprendizagem, é importante treinar os alunos na tarefa de aprender a aprender como propõe [Yang 1998].

4. Um Modelo para o Acompanhamento da Aprendizagem de Programação Apoiado por um STI

O ambiente proposto deverá permitir o registro e acompanhamento do estado mental de cada estudante, em cada tópico das disciplinas de programação, durante todo o curso. Isto possibilitará que professores de turmas subseqüentes possam tomar providências para recuperar alunos com deficiências de aprendizagem, indicando "tratamento" personalizado através de grupos de estudo, aulas de reforço, práticas laboratoriais assistidas, conteúdos adicionais, etc.

O ambiente poderá ser uma ferramenta útil aos coordenadores de curso, capaz de fornecer uma fotografia instantânea do perfil dos futuros egressos, podendo ser utilizado em análises qualitativas, para que sejam tomadas medidas corretivas, servindo assim para melhorar a qualidade dos profissionais formados pela instituição. Apresentam-se a seguir alguns mecanismos que irão, de forma integrada, compor um modelo capaz de acompanhar a aprendizagem de cada aluno.

4.1. Um Modelo Conceitual de Conteúdos Educacionais sobre Programação de Computadores

Os currículos dos cursos da área de Informática geralmente apresentam duas ou três disciplinas de programação por semestre letivo. O "Plano Pedagógico para os cursos de Ciência da Computação segundo as diretrizes curriculares do MEC", possibilita relacionar disciplinas que não exigem pré-requisitos (básicas), que possuem um ou dois níveis de pré-requisitos (intermediárias) e que exigem três ou mais disciplinas básicas/intermediárias (avançadas).

Em muitas disciplinas, os conteúdos estão sobrepostos e o professor acaba revendo conceitos já ensinados ao ensinar uma nova linguagem de programação. O objetivo é analisar os currículos dos cursos de graduação em informática, identificando todas as disciplinas que envolvem programação, e através da análise dos conteúdos programáticos, criar uma base sem redundâncias, capaz de cobrir todos os conceitos necessários para formar um programador completo. A partir desta base, será possível definir um modelo conceitual de conteúdos educacionais da área de Programação de Computadores, utilizando ferramentas adequadas para representação de conhecimento como Mapas Conceituais, Mapas de Informação, Mapas de Hierarquias de Aprendizagem, Mapas Tridimensionais. [Leite 1999] ou Ontologia [Marietto 2000].

Como trabalho inicial, foi desenvolvida a ontologia da disciplina Introdução à Programação (IP-INF), ministrada no primeiro semestre de um curso de graduação em Informática, de uma Instituição de Ensino Superior utilizando a ferramenta Protégé-2000 [Protégé-2000 2002]. Um fragmento desta ontologia pode ser visto na Figura 1.

Ontologias especificam um vocabulário relativo a um determinado domínio. Este vocabulário define os termos (classes, predicados, entidades, propriedades e funções) e as relações entre estes termos, constituindo-se em uma ferramenta poderosa para suportar a especificação e a implementação de sistemas computacionais de qualquer complexidade [Guizzardi 2000].

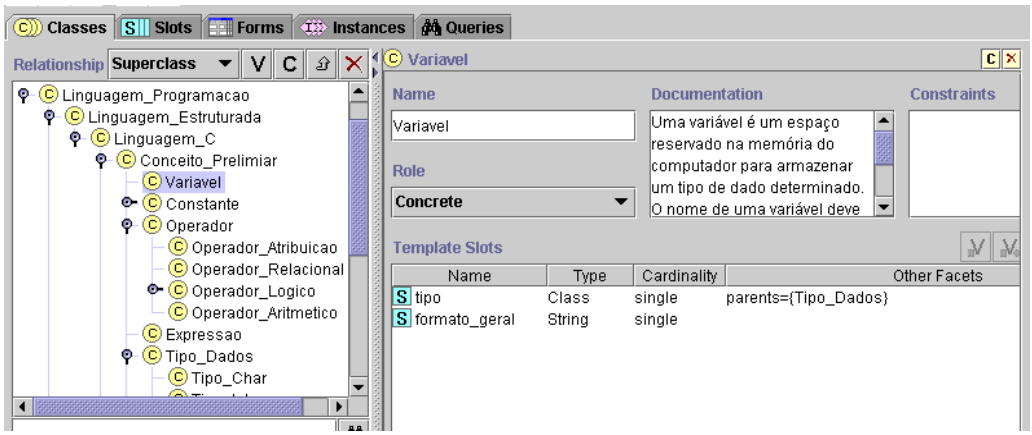

Figura 1. Fragmento da ontologia de linguagem de programação

4.2. Identificando o Estado Mental do Aprendiz

No estudo do processo do ensino-aprendizagem fica evidenciado que é imprescindível considerar o mundo onde o aluno se situa. Este é o ponto de partida para a aprendizagem significativa [Moreira 1999]. É necessário criar um mecanismo para identificar o estado mental de cada aprendiz, relativo ao domínio de conhecimento em questão. Isto pode ser feito através de questionário, entrevista ou mesmo uma avaliação diagnóstica que visa aferir o nível mínimo de conhecimento do aluno.

Como primeira atividade para determinar o que o aluno já sabe elaborou-se um questionário abordando todos os tópicos do conteúdo da disciplina IP-INF, baseado na ontologia criada, no qual o aluno deverá registrar o seu grau de confiança em relação a estes conteúdos. Para cada tópico o aprendiz deverá assinalar valores entre zero e cinco. Quanto mais próximo de cinco, maior a confiança e quanto mais próximo de zero, menor a confiança do aluno sobre seus conhecimentos naquele tópico.

Com este questionário será possível identificar os alunos que já possuem alguma formação em determinados conteúdos, bem como estabelecer o quanto cada aluno acredita saber. Vale ressaltar que o objetivo deste tipo de questionário é o de medir o grau de confiança do aluno naquilo que ele já sabe, e não avaliar o estudante. A Figura 2 mostra uma parte do questionário.

CONCEITO	RESPOSTA
Algoritmos e Programas em C	
1. Algoritmo	
2. Fluxogramas	
3. Estrutura de um Programa em C	
3.1) Cabeçalho de Função	
3.2) Declarações Globais e Locais	
3.3) Diretivas de Pré-processamento	
3.4) Declaração de Variáveis	
3.5) Comentários	
Estruturas de Controle	
1. Comandos de Atribuição	
1.1) Operadores	
1.2) Funções de Biblioteca de C	
2. Comandos Compostos	
3. Comandos Condicionais (*if, if-else*)	

Figura 2. Parte do questionário para levantamento do grau de confiança

O resultado do questionário servirá como ponto de partida para modelar o estado mental de cada aprendiz que será atualizado no andamento da disciplina, a partir das avaliações e registros do professor (veja item 4.3). As avaliações deverão ser elaboradas em termos da ontologia para que seja possível aferir e registrar o que o aprendiz sabe sobre cada tópico.

4.3. Critérios e Meios para a Atualização do Estado Mental do Aprendiz

O acompanhamento da aprendizagem de cada aprendiz é o foco central deste trabalho. Para tal, será preciso criar mecanismos capazes de atualizar o estado mental do aluno, para cada conceito a ser aprendido, em cada disciplina, baseado na ontologia. Desta forma, será necessário estabelecer um critério capaz de aferir com segurança se o aluno aprendeu ou não determinado conceito, para que seu estado mental seja atualizado no STI. Por exemplo, para certificar o aluno no tópico de entrada de dados na disciplina de "Princípios de Desenvolvimento de Algoritmos" (ou com nome similar) pode se estabelecer um conjunto de problemas, com vários níveis de dificuldade que o aluno deverá resolver individualmente, com ou sem o uso de um computador.

Sistemas Automáticos de Avaliação para a certificação em programação poderão ser utilizados, pois possibilitam a aplicação de um número maior de práticas e podem tornar mais ágil a correção de programas, uma vez que no método manual de correção de o professor pode considerar errado um programa que também realiza o que foi pedido no enunciado, mas que está diferente da sua resposta padrão [Kurnia et al. 2001]. Há outros aspectos que podem influenciar a nota final, como a ausência ou exagero de comentários nos programas, a forma como o aprendiz apresenta o programa (estética, endentação), os nomes de variáveis utilizados e até mesmo o humor de quem corrige os programas.

Ferramentas automáticas de avaliação tornam-se ainda mais necessárias quando se trabalha com grupos grandes e podem ser utilizadas não como um único critério, mas como um critério adicional que torne o ato de avaliar mais justo para todas as partes.

4.4. Arquitetura de um STI para Avaliação e Acompanhamento Contínuo da Aprendizagem

O acompanhamento contínuo da aprendizagem e o tratamento personalizado a cada estudante, principalmente em turmas com um número elevado de alunos, só será possível se for apoiado por sistemas inteligentes, ou seja, auxiliado por computadores.

Os processos de aprendizagem auxiliados por computador requerem a implementação de sistemas complexos, dinâmicos e adaptativos. Estes sistemas devem ser capazes de se adequar ao estado mental de cada estudante, num dado instante, a partir de um modelo histórico e do desempenho instantâneo do aprendiz. Atualmente, os STIs são os que melhor respondem a estas exigências [Leite 1999].

A arquitetura do STI que irá apoiar o ambiente de acompanhamento e avaliação contínua de aprendizagem tem como base cinco módulos, a saber: domínio, estudante, especialista, tutor e comunicação conforme Figura 3.

A seguir, apresenta-se a descrição inicial de alguns serviços a serem realizados pelo núcleo do STI em seus módulos principais.

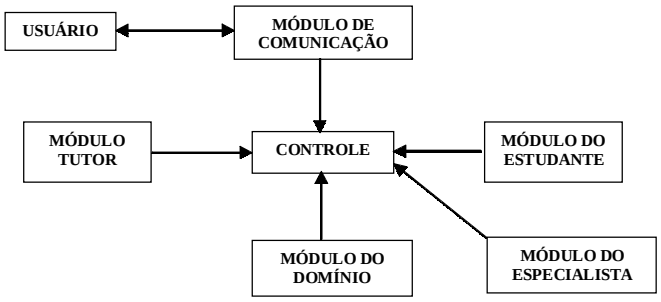

Figura 3. Proposta de arquitetura para o STI

Módulo do Domínio: O domínio a ser tutorado pelo STI deverá abranger tópicos de programação, do básico ao avançado, envolvendo diversos paradigmas de programação: procedural, orientada a objetos, concorrente, distribuída, em lógica, etc. Estes assuntos estarão dispostos de maneira hierárquica, descrevendo os requisitos e de alguma forma indicando as disciplinas em que cada item será abordado. A ontologia que está sendo definida para o domínio de programação conforme descrito no item 4.1 servirá de base para a alimentação deste módulo.

Módulo do Estudante: Deverá descrever o estado mental do aprendiz, ou seja, armazenar informações referentes ao que o estudante conhece sobre os vários tópicos de programação. O questionário de identificação do grau de confiança do aprendiz descrito no item 4.1 será utilizado como fonte inicial para compor o modelo do estudante. Posteriormente, outros mecanismos poderão ser utilizados. [Cury 1996] propõe um conjunto de ferramentas de autoria para a modelagem dos estereótipos de aprendizes.

Módulo do Especialista: Deverá conter um modelo, considerando-se as habilidades e competências que o especialista possui nos conteúdos especificados no módulo do domínio**. E**ste modelo servirá como parâmetro de comparação com o estado mental do aprendiz que deverá atingi-lo para ser considerado um perito no assunto. Para cada item do módulo do domínio serão especificados níveis hierárquicos de conhecimento. A Figura 4 mostra estes níveis.

Na base da pirâmide encontram-se os conhecimentos considerados mínimos para o aprendizado. No nível de conhecimento complementar, o aluno deverá ser capaz de generalizar os conhecimentos adquiridos no nível anterior e, no topo da pirâmide o aluno deverá ser capaz de abstrair os diversos conhecimentos adquiridos, aplicando-os em situações problema diferentes.

Figura 4. Níveis de Conhecimento

As medidas educacionais utilizadas no sistema atual (com média 5.0) possibilitam que o aluno seja promovido sem os conhecimentos mínimos, uma vez que há um subdimensionamento no processo de avaliação.

Módulo do Tutor: Conterá o plano pedagógico das disciplinas, sendo responsável pela decisão sobre como e que atividades pedagógicas serão apresentadas, sempre interagindo com o módulo do estudante [Cury 1996]. O STI funcionará como um monitor de laboratório, sempre presente para observar o aprendizado do estudante.

5. Aplicação da Metodologia

Com o objetivo de validar a proposta descrita acima, iniciou-se uma coleta de dados através do acompanhamento de alunos de disciplinas de Programação em cursos de Informática de duas Instituições de Ensino Superior, sendo as disciplinas Introdução à Programação (IP-INST1) do primeiro semestre do curso de Engenharia de Computação da Instituição 1, designada por INST-1, e Estruturas de Dados (ED-INST2) do segundo ano dos cursos de Ciência da Computação e Sistemas de Informação da Instituição 2, designada por INST-2. A seguir o detalhamento destas experiências iniciais.

5.1. Aplicação do Questionário Inicial na Disciplina IP-INST1

O questionário (conforme descrito no item 4.2) foi aplicado para 38 alunos de duas turmas, T1 e T3, do primeiro semestre da disciplina IP-INST1. Como são alunos ingressantes, nada se esperava destes em termos dos conteúdos abordados na disciplina. No entanto, alunos que cursaram segundo grau técnico, que fizeram cursos de informática ou que são autodidatas, efetivamente poderiam apresentar conhecimentos sobre determinados conteúdos. Na Tabela 1, apresentamos um resumo dos resultados das 73 questões, agrupadas em tópicos.

Tabela 1. Resultados do questionário aplicado na disciplina IP-INST1

ITEM	Quantidade de respostas diferentes de zero		Média das respostas (valores entre 0 e 5)	
	T1	T3	T1	T3
Conceitos Básicos de Programação	22	16	**1.75**	**1.33**
Algoritmos e Programas em C	8	6	**0.56**	**0.57**
Estruturas de Controle	8	4	**0.59**	**0.37**
Comandos de Entrada e Saída	6	6	**0.47**	**0.39**
Variáveis Estruturadas	4	3	**0.30**	**0.17**
Subprogramação	4	5	**0.35**	**0.26**
Ponteiros	2	3	**0.30**	**0.17**
Noções de Estruturas de Dados	4	3	**0.39**	**0.31**

Com as médias obtidas nas duas turmas, é possível afirmar que boa parte dos alunos não acredita saber quase nada sobre os conteúdos abordados na disciplina, exceto no primeiro item (conceitos básicos de programação) que obteve respectivamente médias 1.75 e 1.33 para as turmas T1 e T3, de um máximo de 5.0. Uma outra leitura pode ser feita pela coluna "Quantidade de Respostas diferentes de zero" é que cerca de 3 a 8 alunos expressaram ter algum conhecimento (entre 1 e 5) sobre alguns tópicos da disciplina.

5.2. Aplicação do Questionário Inicial na Disciplina ED-INST2

Com o objetivo de medir o grau de confiança dos alunos que já passaram por um processo inicial de formação em Programação aplicou-se um questionário similar ao da disciplina IP-INST1, mas adaptado à realidade da disciplina cursada no primeiro ano da Instituição 2. Mesmo os alunos tendo passado pela disciplina introdutória, não existem registros do que efetivamente cada aluno sabe ou não sabe, sendo impossível estabelecer um programa de recuperação ou nivelamento para os alunos com deficiências em determinados conteúdos. O questionário na Instituição 2 foi aplicado em cinco turmas (c2a, c2b, c2c, c2d. s2a), totalizando 259 alunos. Os dados de cada aluno, em cada item, estão armazenados em um banco de dados o que permitirá a atualização do estado mental de cada um deles, à medida que as verificações de aprendizagem forem acontecendo.

A Tabela 2 apresenta um quadro resumo detalhando o percentual por tipo resposta (valores de 0 a 5) e por turma, abrangendo um universo de 22792 respostas (= 259 alunos x 88 questões). Algumas leituras gerais podem ser feitas, como por exemplo, o fato de que, pelos resultados, os alunos da turma "s2a" possuem grau de confiança menor, pois em 23.5% (10.2%+ 11.3%) destes responderam grau 0 ou 1. No entanto, é possível observar um equilíbrio nas outras quatro turmas, uma vez que os valores percentuais de cada coluna são bem próximos em todas as turmas.

Tabela 2. Percentual por tipo de resposta (grau) em cada turma do questionário aplicado na disciplina ED-INST2

TURMA	TOTAL ALUNOS	QUANT. GRAU 0	QUANT. GRAU 1	QUANT. GRAU 2	QUANT. GRAU 3	QUANT. GRAU 4	QUANT. GRAU 5
c2a	71	5,4%	5,8%	14,7%	27,6%	26,8%	19,7%
c2b	68	4,2%	8,5%	19,2%	25,9%	23,1%	19,1%
c2c	54	4,7%	8,3%	18,4%	26,3%	24,8%	17,6%
c2d	27	5,6%	5,5%	16,8%	25,8%	22,3%	24,0%
s2a	39	10,2%	11,3%	23,0%	26,2%	17,2%	12,0%
TOTAL	**259**	**5,7%**	**7,8%**	**18,1%**	**26,5%**	**23,5%**	**18,4%**

Para o efetivo acompanhamento da aprendizagem, o que importa de fato são as análises individuais das respostas de cada aluno. No questionário individual, podem-se identificar os itens em que o aluno possui grau de confiança baixo e requer ajuda, seja pelo registro do valor (de 0 a 5), seja por comentários do tipo "eu odeio este conteúdo".

No entanto, análises por grupos de conceitos podem ajudar a identificar os itens do programa em que os alunos, no conjunto, apresentam maior dificuldade, ou seja, menor grau de confiança. Com isto, o professor pode indicar medidas de recuperação através de aulas de revisão, criação de grupos de estudo, exercícios extras etc. A Tabela 3 mostra um quadro resumo por conceito, detalhando as médias por turma.É importante dizer que, por questões de espaço, dos 88 subitens do questionário, apenas alguns itens gerais, considerados mais importantes, são apresentados. No entanto, os dados de cada um dos subitens estão armazenados em banco de dados e podem ser acessados quando necessário. Dentre outras análises, observa-se que na coluna "Média Geral", o Item "10 - Recursividade" obteve média 1.9, sendo que na turma "s2a" este valor foi de apenas

0.9, indicando que este conceito precisa ser trabalhado novamente. O tópico "9 - Passagem de Parâmetros" e "6 - Cadeia de Caracteres (string)" também são itens com média relativamente baixa, respectivamente 2.4 e 2.7.

Tabela 3. Média por grupos de conceitos

CONTEÚDO	MÉDIAS					
	C2A	C2B	C2C	C2D	S2A	GERAL
1- Entrada de Dados	3,6	3,2	3,6	3,9	3,3	3,5
2 – Saída de Dados	4,2	4,0	4,1	4,2	4,2	4,1
3- Estruturas de Seleção	3,7	3,6	3,5	3,8	3,2	3,6
4 – Estruturas de Repetição	3,8	3,8	3,6	3,7	3,2	3,6
5 – Cadeia de Caracteres (String)	2,9	2,8	2,7	2,9	2,3	2,7
6 – Estruturas de Dados (Vetores)	3,1	3,3	3,2	2,9	2,6	3,0
7 – Uso de funções pré-definidas	3,2	2,8	2,9	3,3	2,2	2,9
8 – Definição de funções em C	3,5	3,1	3,2	3,4	2,4	3,1
9 – Passagem de Parâmetros	2,8	2,6	2,5	2,2	1,9	2,4
10 – Recursividade	2,1	2,2	2,0	2,4	0,9	1,9
11 – Compilador C (Tclite, Turbo C, etc)	2,5	2,4	2,7	3,0	2,2	2,6
12 – Domínio de Lógica de programação	3,3	3,1	3,0	3,1	2,5	3,0
MÉDIA GERAL	**3,3**	**3,2**	**3,1**	**3,4**	**2,7**	**3,1**

A Tabela 4 apresenta os mesmos itens (conteúdos), detalhados por faixa de grau de confiança. Cada coluna mostra a quantidade de alunos que responderam 0 ou 1, 2 ou 3, e 4 ou 5, para cada tópico. Os dados estão ordenados em ordem decrescente de "grau 0-1" (primeira coluna), o que permite identificar facilmente quais foram os itens que tiveram maior quantidade de respostas 0-1, ou seja, itens em que os alunos apresentam menor grau de confiança. Por exemplo, o item "10 – Recursividade" aparece com 89 respostas 0 ou 1, ou seja, 34,3%. No entanto, há alguns dados que pode gerar informações falsas: a Tabela 4 aponta que os alunos possuem maior dificuldade no item "7 – Uso de funções pré-definidas" do que no item "9 – Passagem de Parâmetros" ou no item "8 – Definição de funções em C", o que na prática não é absolutamente verdade. Vale lembrar que grau de confiança é o sentimento do aluno e não propriamente a realidade do seu saber. A confirmação disto poderá ser feita com a aplicação de verificações de aprendizagem.

6. Considerações Finais

O modelo proposto para a avaliação e acompanhamento contínuo da aprendizagem tem como objetivo contribuir para um maior rendimento dos estudantes de linguagem de programação, incentivando-os a redescobrirem seus potenciais e conseqüentemente melhorar os seus processos de aprendizagem.

A validação do modelo está sendo realizada com base nos dados obtidos com a aplicação de questionários em turmas de Programação de primeiro e segundo ano em duas Instituições. Posteriormente, à medida que ferramentas de apoio forem

construídas, pretende-se aplicar a metodologia de maneira integrada e contínua, iniciando com disciplinas introdutórias e avançando por disciplinas subseqüentes.

Como trabalhos futuros vislumbra-se a possibilidade de validar o modelo com outras áreas de conhecimento, não ficando restrito ao domínio aqui proposto e que o acompanhamento do estudante seja feito de forma automatizada.

Tabela 4. Média por grupos de Conceitos nas turmas da disciplina ED-INST2

ITENS DE CONTEÚDO	QUANTIDADE DE ALUNOS		
	GRAU 0-1	GRAU 2-3	GRAU 4-5
10 – Recursividade	89	131	39
7 - Uso de funções pré-definidas	42	133	84
11 – Compilador C (Tclite, Turbo C, etc)	41	132	86
9 – Passagem de Parâmetros	35	158	66
12 - Domínio de Lógica de programação	25	111	123
8 – Definição de funções em C	22	109	128
6 – Estruturas de Dados básicas (Vetores)	20	125	114
5 - Cadeia de Caracteres (String)	19	141	99
4 – Estruturas de Repetição	10	77	172
1 - Entrada de Dados	3	95	161
3 – Estruturas de Seleção	2	79	178
2 - Saída de Dados	1	48	210
PERCENTUAL GERAL	9,6%	43,6%	46,8%

Referências

Brown, M. (1988). Exploring algorithms using BALSA-II. IEEE Computer. V 21 No. 5.

Brown, M.H. (1991). Zeus: A System for Algorithm Animation and Multi-view Editing, Research Report n. 75, DEC Systems Research Center, Palo Alto, CA.

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

Cury, Davidson (1996). Flama: Ferramentas e Linguagem de Autoria para a Modelagem da Aprendizagem. Tese de Doutorado. São José dos Campos: Instituto Tecnológico da Aeronáutica.

Gomes, A. J. Ambiente de Suporte à Aprendizagem de Conceitos Básicos de Programação. Dissertação de Mestrado. Universidade de Coimbra, 2000.

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, Chile.

Guizzardi, G. (2000) Uma Abordagem Metodológica de Desenvolvimento para e com Reuso, Baseada em Ontologias Formais de Domínio. Dissertação de Mestrado. Universidade Federal do Espírito Santo.

Jandl, Peter Jr. e Zuchini, Márcio H. (1999). IC: Um Interpretador de Linguagem C. IN Projeções - USF, Bragança Paulista, V. 17, pp. 101-112.

Kurnia, A., Lim, A e Cheang, B. (2001) Online Judge. Computer & Education, v. 36 p. 299-315.

Leite, A. de Sá (1999). Um Modelo de Sistema Educativo Cognitivista Baseado em Tutoria Inteligente Adaptativa Via Aderência Conceitual. Tese de Doutorado. São José dos Campos: Instituto Tecnológico da Aeronáutica.

Marietto, M. G. B. (2000). Definição Dinâmica de Estratégias Instrucionais em Sistemas de Tutoria Inteligente: Uma Abordagem Multiagentes na WWW. Tese de Doutorado. São José dos Campos: Instituto Tecnológico da Aeronáutica.

Moreira, Marco A. (1999). Teorias de Aprendizagem. São Paulo: EPU.

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

Protégé-2000 (2002). Protégé-2000 home page: http://protege.stanford.edu/index.html.

Rezende, P.J. e Garcia, I.C. (1995). Astral: Animação Gráfica de Algoritmos e Estruturas de Dados - Uma Abordagem Construtiva, VIII Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens, 317-318.

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.

Tobar, Carlos Miguel et al. (2001). Uma Arquitetura de Ambiente Colaborativo para o Aprendizado de Programação. XII Simpósio Brasileiro de Informática na Educação, Vitória - ES.

Yang, Nae-Dong (1998). Exploring a New Role for Teachers: Promoting Learner Autonomy. System, v. 26 p. 127-135.