Utilizando o Método Clínico Piagetiano para Acompanhar a Aprendizagem de Programação

Francisco Araújo de Almeida Neto[1], Thais Helena Chaves de Castro[2], Alberto Nogueira de Castro Júnior[2,1]

[1]Programa de Pós-Graduação em Informática – Universidade Federal do Amazonas (UFAM)

[2]Departamento de Ciência da Computação – Universidade Federal do Amazonas (UFAM)

Av. Gal. Rodrigo O. J. Ramos, 3000-Setor Sul, Bl. M – 69077-300– Manaus – AM – Brasil

{faan,thais,albertoc}@dcc.ufam.edu.br

Abstract. *This paper describes an on-going project that aims to investigate the process of knowledge construction in programming. From the proposal of using Piaget´s Clinical Method for analyzing learning difficulties in students from introductory courses on programming, a pilot-study was carried out and it has made clear the need for a software tool for supporting recording, structuring and analysis of artifacts (programs) produced by students. After describe the development of that tool, we discuss its use with respect to the sort of monitoring proposed.*

Resumo. *Este artigo descreve um projeto em andamento que busca investigar o processo de construção do conhecimento em programação. A partir da proposta de utilizar o Método Clínico Piagetiano para analisar os problemas de aprendizagem dos estudantes de disciplinas introdutórias de programação, foi realizado um estudo-piloto onde ficou evidente a necessidade de uma ferramenta para registro, organização e análise dos artefatos (programas) produzidos pelos estudantes. Após a descrição do desenvolvimento de tal ferramenta, discutimos a utilização da mesma com respeito ao tipo de acompanhamento proposto.*

Palavras-chave: aprendizagem de programação, ambientes telemáticos, método clínico piagetiano

1. Introdução

A aprendizagem de programação pressupõe a transposição de alguns patamares de expertise até que o sujeito seja um programador experiente. Esses patamares, embora cogitados por muitos pesquisadores, ainda não são conhecidos, e por isso, ensinar programação não poderia seguir uma metodologia linear. Cada curso de computação opta por uma seqüência diferente de estudo dos paradigmas de programação em seus currículos, o que é usualmente confirmado por diretrizes curriculares da área de computação [Engel e Roberts, 2001]. No entanto, praticamente não há, na literatura,

resultados experimentais que comprovem a eficiência de alguma metodologia. Isso sugere que de fato não conhecemos os processos cognitivos envolvidos na aprendizagem de programação.

A despeito da inexistência de um consenso sobre a abordagem mais adequada ao ensino de programação, alguns aspectos do tema são amplamente aceitos, por exemplo: A importância dessa competência na formação do profissional de computação; a diversidade de caminhos na construção das habilidades necessárias ao desenvolvimento da *expertise* na atividade; e não menos importante, as inúmeras dificuldades vividas por grande parte dos estudantes nas disciplinas introdutórias de programação.

A partir do pressuposto que um dos principais componentes na aprendizagem de programação é a organização das habilidades para a resolução de problemas, elemento comum à construção de conhecimento em qualquer outro domínio, surge a necessidade de investigar como tais habilidades são construídas, o que envolve identificar e entender as dificuldades encontradas pelos estudantes de programação. Uma vez que estamos nesse caso tratando da formação do que, segundo Piaget, são estruturas cognitivas avançadas, é possível utilizarmos o Método Clínico proposto por ele, para realizar tal análise.

Após um estudo-piloto com a utilização do método, ficou evidente a necessidade de suporte tecnológico para o registro, organização e análise dos artefatos (programas) produzidos pelos estudantes. Este trabalho descreve o contexto do desenvolvimento de uma ferramenta de software com tal propósito, bem como os cenários de sua utilização para apoiar o tipo de acompanhamento proposto. A próxima seção descreve o estudo-piloto desenvolvido e os resultados encontrados. Na Seção 3, descrevemos aspectos relevantes do desenvolvimento da ferramenta. Na Seção 4 ilustramos o uso da mesma, e na última seção discutimos o cenário relacionado à aplicação do método utilizando essa ferramenta de apoio.

2. Analisando a Aprendizagem de Programação

No primeiro semestre acadêmico de 2004 foi realizado um experimento prospectivo com três turmas de estudantes iniciantes em programação, graduandos do Bacharelado em Ciência da Computação e do curso de Engenharia da Computação em nossa IFES. Foram utilizadas duas disciplinas introdutórias e utilizados alguns instrumentos de avaliação qualitativa, como questionários e desafios de lógica. Conforme relatado em [Castro et al, 2004], esse experimento forneceu evidências de dificuldades de aprendizagem em itens específicos do currículo da disciplina de Introdução à Programação.

Uma vez conhecidos alguns dos itens de maior dificuldade para os estudantes, foi elaborado um estudo-piloto, utilizando o método clínico proposto e utilizado amplamente por Piaget [Delval, 2002], para aprofundar a análise de tais elementos. O principal motivador foi o fato de que somente uma avaliação com questionários não era suficiente para inferir a verdadeira dificuldade na utilização de um conceito, pois a programação é uma tarefa muito abstrata de resolução de problemas, onde os próprios estudantes desconhecem os processos mentais que utilizam.

O método clínico consiste em entrevistas individuais com os estudantes durante a resolução de um problema elaborado segundo os critérios de resolução que se deseja

estudar. Por exemplo, ao estudar as construções por meio de cartas [Piaget, 1978], Piaget apresentava à criança entrevistada um maço de cartas de baralho, pedindo-lhe para construir uma "casa". Normalmente a criança apresentava alguma dificuldade inicial. Então, o condutor do experimento começava pelo teto da casa, formando um "T". Após algumas tentativas da criança eram feitos questionamentos quanto a noções de equilíbrio, procurando antever situações, com a retirada ou afastamento de uma carta.

A situação descrita acima e as demais utilizadas por Piaget, envolviam manipulações concretas, embora envolvessem muitos processos de raciocínio lógico, indutivo e de representação. No caso de o objeto manipulado ser algo abstrato também, como programas de computador, é necessário um ajuste na aplicação do método. Por isso, o planejamento do estudo-piloto envolveu algumas adequações no método clínico original.

Em nosso caso, como havia duas turmas da disciplina Introdução à Computação, três problemas diferentes e com o mesmo grau de complexidade eram trabalhados a cada semana. O terceiro problema era fornecido a uma terceira turma (grupo de controle), formada por cinco estudantes de cada uma das outras turmas, escolhidos por não terem experiência anterior com programação.

O estudo piloto consistiu de observações externas durante as etapas de codificação dos programas, observações internas (anotadas pelos próprios estudantes) antes e depois de resolução dos problemas, e de entrevistas individuais. Os problemas de laboratório eram elaborados de acordo com a complexidade dos problemas trabalhados durante as aulas teóricas e se utilizava o método do "problema do dia", ou seja, os estudantes resolviam um problema por dia durante suas aulas práticas, em laboratório. Todos os procedimentos utilizados foram registrados em um ambiente computacional.

Quanto às observações dos sujeitos (estudantes), logo depois da leitura do problema do dia, cada estudante preenchia uma ficha simplificada, fornecendo suas impressões iniciais sobre o grau de dificuldade. Imediatamente após o preenchimento da ficha, eles iniciavam a resolução do problema. Posteriormente, ao finalizarem a codificação de suas soluções, escreviam novamente algumas observações, mas desta vez contendo suas impressões sobre o grau de dificuldade para resolução do problema e o quanto essas impressões finais se aproximaram ou se distanciaram de suas impressões iniciais.

As observações externas eram realizadas nas três turmas. Nas turmas regulares, os próprios monitores (um para cada turma) observavam a turma de maneira geral e relatavam o que lhes chamava a atenção, como possíveis soluções interessantes ou comportamentos diferentes durante as sessões de laboratório. Se, ao ler os relatórios dos monitores, o professor notasse algo interessante, poderia chamar aquele estudante para uma entrevista. Na turma extra (grupo de controle), além do monitor, havia cinco observadores, dos quais três eram estudantes de mestrado e dois finalistas do Bacharelado em Ciência da Computação. Cada um ficava responsável por dois estudantes. Esses observadores anotavam tudo o que cada estudante fazia, como tentativas mal sucedidas, distrações, consultas a materiais ou colegas, tempo de resolução, etc.

Um dia depois do laboratório, um ou dois estudantes do grupo de controle eram chamados para uma entrevista com a professora da disciplina para falarem sobre os processos que utilizaram para elaboração de suas soluções. Na Tabela 1, construída com o material registrado no ambiente para trabalho colaborativo utilizado, está um exemplo de observação de um estudante, a observação do assistente de pesquisa sobre esse estudante e a codificação.

Tabela 1 - Um exemplo de problema trabalhado

Descrição do Problema	Em um pronto socorro particular, como o da UNIMED, assim que um paciente chega para ser consultado recebe uma senha com um número de atendimento. Existem sempre 3 médicos disponíveis, para os quais são enviados pacientes de acordo com a quantidade de pacientes à espera por ele, de forma que o medico que possuir o menor número de pacientes à espera recebe o próximo. Assim, podemos definir, utilizando tuplas, a seguinte entrada: medicos_disponiveis (("dr. A",4,23),("dr. B",1,13),("dr. C",3,27)), onde o 2o. termo de cada tupla refere-se ao número de pacientes na fila daquele médico e o 3o. refere-se à última senha que aquele médico recebeu. Baseado nesta entrada, faça um script em Haskell para, dada a entrada de um paciente no pronto socorro, forneça uma senha para ele e indique para qual médico deve ir. Obs.: Lembre-se que o critério para direcionamento de pacientes é pelo menor número de atendimentos agendados para um médico.
Observações Iniciais (estudante)	1 - Estou certo que conseguirei resolver 2 - Já resolvi problemas semelhantes, por isso soube por onde começar

Codificação	`module PS28_de_agosto where`
	
	`medicos_disponiveis(("A",pa1,sen1),("B",pa2,sen2),("C",pa3,sen3))`
	`= (med,pac,sen)`
	`  where`
	`  med = if ((pa1<=pa2) && (pa1<=pa3))`
	`  then"A"`
	`  else if ((pa2<pa1) && (pa2<=pa3))`
	`  then "B" else "C"`
	`  pac = if (med == "A")`
	`  then (pa1 + 1)`
	`  else if (med == "B")`
	`  then (pa2 + 1) else (pa3 + 1)`
	`  sen = if (med == "A")`
	`  then (sen1+pa1+1)`
	`  else if (med == "B")`
	`  then (sen2+pa2+1) else (sen3+pa3+1)`

Observações Finais (estudante)	No momento em que li o problema não o entendi bem como fazer, então o li de novo atentamente, logo entendi o que o problema queria; após algumas tentativas frustradas, consegui resolver. Primeiro, sabendo qual o médico o paciente deveria ir, vi que o problema praticamente já estava quase resolvido, precisando apenas do lugar dele na lista de espera e de sua senha, qual o doutor que tivesse menos numero de pacientes na fila, para esse irá o nosso proximo paciente, a senha dele será o numero da senha ki o medico já recebeu mais as senhas daqueles pacientes que já estavam na fila. Assim resolvido o problema!
Observações Externas	16:16 – Lendo o enunciado do problema. O estudante já havia começado a desenvolver o exercício em casa. 16:21 – Parou para ouvir explicações da monitora. 16:23 – Perguntou ao colega Kaio se precisa comentar. 16:25 – Consultou ao colega Kaio para esclarecimento de duvidas sobre o problema. 16:27 – Executou solução vendo no código os erros. 16:34 – Teste do Código OK. 16:35 – Testando com outros valores e o teste deu OK! 16:39 – Pede ajuda a monitora sobre duvidas sobre problema. 16:41 – Ajuda o cologa Kaio a tirar duvidas do problema.

	16:45 – Testa seu código com últimos resultados pra verificar corretude e tudo dá OK! 17:46 – Terminando solução e respondendo questionário.

2.1. Resultados Experimentais

Com o término da disciplina, foi realizada uma análise comparativa entre os comentários dos observadores, os scripts dos estudantes, scripts padrões codificados pelos professores e as observações dos estudantes.

Percebeu-se que, apesar de a ênfase do curso ser em resolução de problemas, os estudantes, ao elaborarem uma solução, pulavam partes importantes do planejamento, recaindo frequentemente nos mesmos problemas por busca de soluções por tentativa e erro. Além disso, não foi possível perceber muitas das dificuldades na apreensão dos conceitos, pois os estudantes ficavam reescrevendo os mesmos scripts e somente ficava registrada a versão final, com comentários finais. Portanto, mesmo que um estudante fosse registrando seus comentários conforme fosse construindo uma versão da solução, a análise seria limitada uma vez que não estariam disponíveis as versões preliminares desenvolvidas.

Os procedimentos experimentais utilizados poderiam ser muito melhor aproveitados se houvesse uma ferramenta computacional que possibilitasse a análise "interna" dos artefatos produzidos durante experimento, ou seja, a análise da evolução nos programas construídos pelos estudantes, capturando desde modificações significativas no código até pequenas alterações aparentemente sem muita importância.

Conforme dispõem os princípios da Engenharia da Usabilidade e da Engenharia de Software, qualquer ferramenta de software deve apoiar a realização de alguma tarefa relevante, sendo essa a própria razão de sua existência. Em nosso caso, uma demanda bem clara conduziu ao desenvolvimento da ferramenta descrita nas seções seguintes.

3. AAEP – Um Ambiente para Acompanhamento e Análise de Programas

A análise dos resultados do experimento descrito na seção anterior resultou na concepção e desenvolvimento de uma ferramenta de acompanhamento das soluções dos estudantes, com possibilidade de, sempre que o professor julgar necessário, resgatar uma comparação entre duas de quaisquer versões de um mesmo estudante.

O ambiente foi projetado para atender essencialmente às demandas do professor. Portanto, as funcionalidades de gerência, como cadastro de usuários, cadastro de problemas e a análise das soluções é de acesso restrito ao mesmo. As outras funcionalidades, como elaboração, edição e teste de código-fonte de programas, associados a comentários caracterizando cada versão, podem ser realizadas tanto por estudantes quanto por professores.

Conforme sugere o conjunto de funcionalidades enunciado no parágrafo anterior, a ferramenta incorpora três categorias de serviços: a gerência de problemas e de usuários, a operação de um interpretador da linguagem de programação utilizada, e um controle de versões dos programas gerados. Como já existem ferramentas confiáveis e de código-aberto para realizar o controle de versões e o interpretador utilizado deveria ser aquele utilizado na disciplina de programação considerada, o desenvolvimento da ferramenta teve como elemento central a construção de uma camada para gerência de

usuários e problemas, e de estruturas para a agregação de serviços externos. A Figura 1 apresenta o diagrama de componentes para a ferramenta.

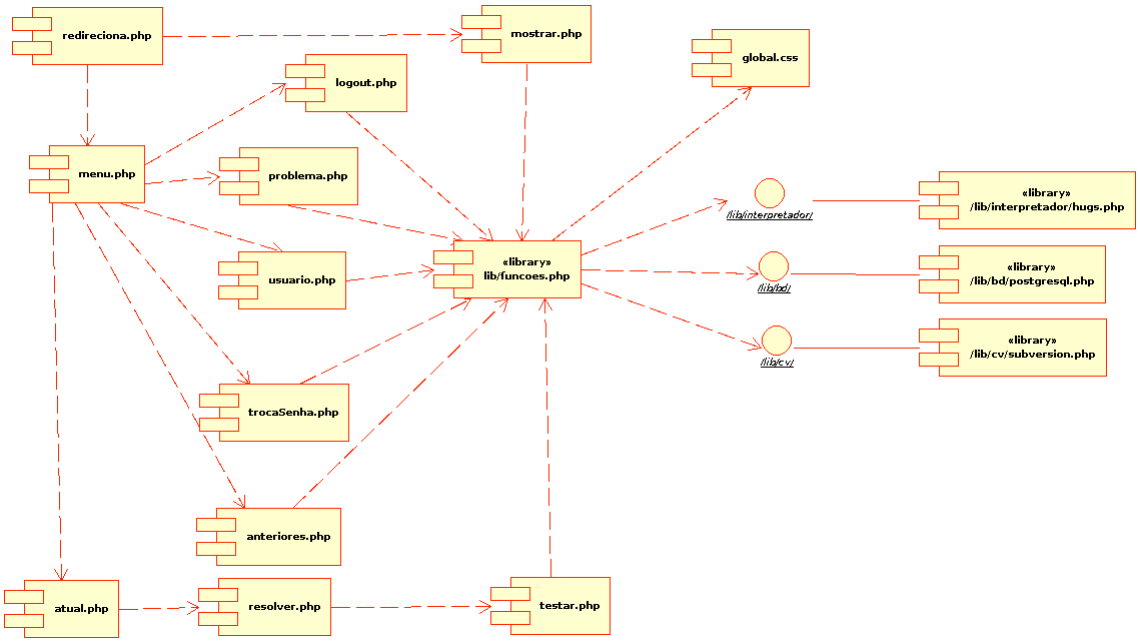

Figura 1 - Diagrama de Componentes do AAEP

A ferramenta foi desenvolvida utilizando plataforma WEB e linguagem de programação PHP. Tal escolha apresenta duas vantagens imediatas: a primeira é a possibilidade de um ciclo rápido de desenvolvimento e testes; e a segunda é que, uma vez que os estudantes já são familiarizados com o ambiente da Internet, torna-se muito mais fácil e intuitivo o aprendizado e utilização do AAEP. Em adição a essas características, a estrutura cliente-servidor também facilita o uso tanto nos laboratórios, no contexto de aplicação do método clínico piagetiano apresentado, quanto em outros cenários, onde ações não presenciais poderão ser incorporadas.

Conforme os estudantes desenvolvem seus programas e testam seu funcionamento, o AAEP vai registrando os estágios intermediários de cada solução em um sistema de Controle de Versões, estabelecendo uma ligação com o Banco de Dados que organiza as contas dos usuários e o conjunto de problemas cadastrados. O AAEP foi criado com uma API para servir de interface entre ele, o Banco de Dados utilizado, o sistema de Controle de Versões usado e o interpretador onde as soluções dos estudantes são testadas. Ao seguir essa abordagem, a aplicação não ficou dependente de nenhum software ou versão específicos. No momento o Banco de Dados utilizados é o PostgreSQL, o Controle de Versões é o Subversion e a ferramenta utilizada para testar as soluções é o interpretador Hugs para a linguagem Haskell, visto que é nessa linguagem que os problemas atualmente propostos são desenvolvidos. Na próxima seção é ilustrado como o sistema está sendo utilizado.

4. Cenários de Uso do AAEP

Conforme indicado na seção anterior, o objetivo que orientou as decisões de projeto do AAEP foi o apoio ao registro, organização e acompanhamento dos programas gerados em disciplinas introdutórias de programação. O professor é o articulador das ações no ambiente, ficando responsável pela gerência de usuários e de problemas. A Figura 2 mostra um *snapshot* da ferramenta em ação referente à gerência de usuários, apresentada na área de trabalho principal da ferramenta. À esquerda da área de trabalho, encontra-se os *links* para os demais serviços.

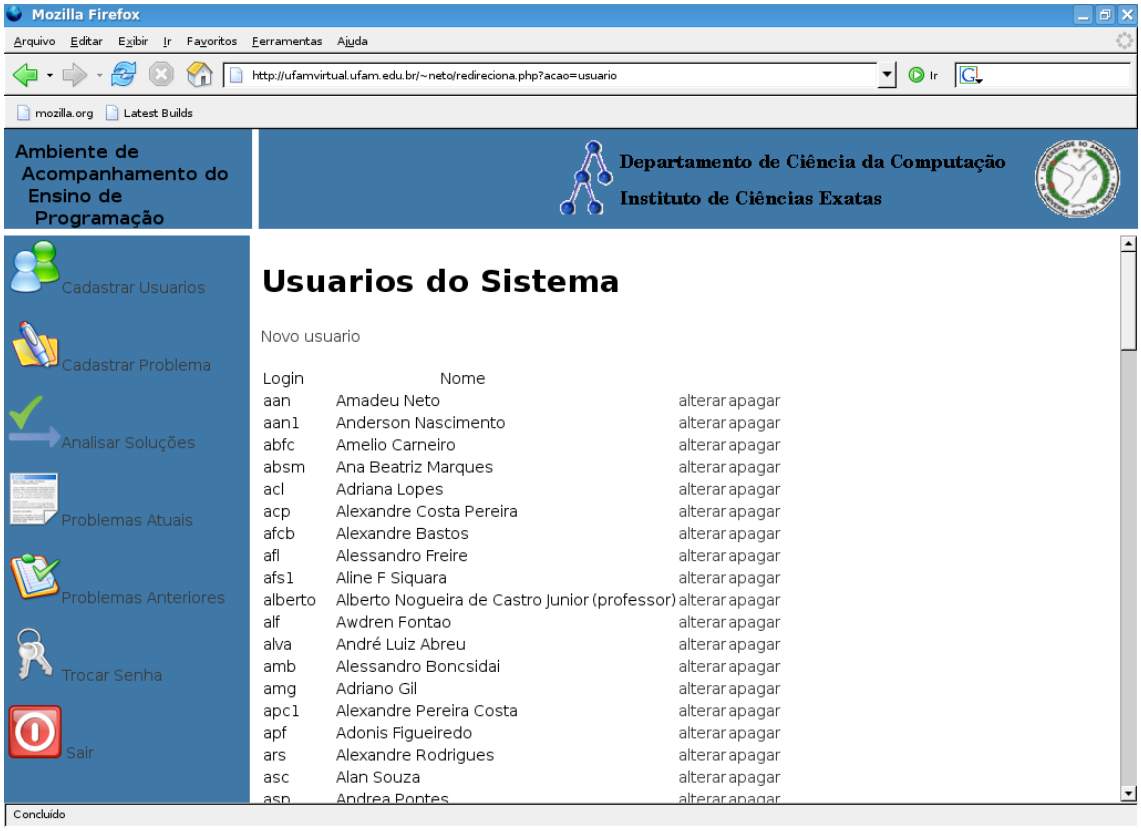

Figura 2 – Gerência de usuários no AAEP

O cadastramento de problemas através de uma janela similar àquela apresentada na Figura 2, exceto pelo fato que uma data-limite para submissão de soluções é também apresentada.

A Figura 3 ilustra a situação onde um problema específico está sendo trabalhado. O nome e a descrição do problema, previamente informados pelo professor, são exibidos ao usuário, que deve escrever as soluções (programas) que podem vir acompanhadas de comentários qualificadores.

Uma vez que a primeira versão da solução tenha sido informada, o usuário pode iniciar o ciclo de testes de sua solução, situação ilustrada na Figura 4. Ao verificar que o comportamento observado difere do planejado, o usuário retorna à situação apresentada na Figura 3 para proceder às correções devidas no código que deve também ser submetido aos testes apropriados.

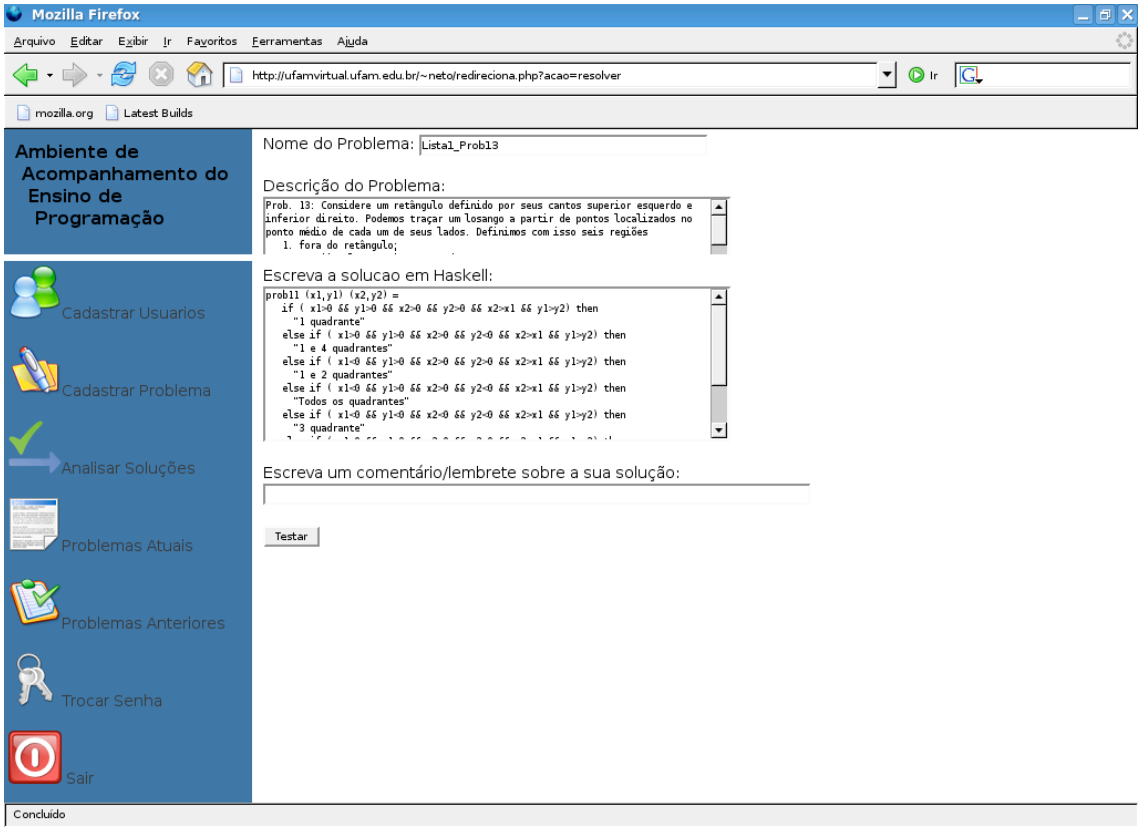

Figura 3 – Elaborando um programa no AAEP

É interessante observar que em sua implementação atual, o AAEP pode ser utilizado em contextos distintos. O cenário inicial é aquele caracterizado pelo estudo-piloto, onde um problema é apresentado a cada nova sessão no laboratório, e as diferentes versões do programa produzidas durante aquela sessão, serão analisadas pelos observadores. Um cenário alternativo, que já está sendo aplicado, é a utilização da ferramenta para o desenvolvimento de trabalhos não supervisionados (listas de exercícios), onde os estudantes utilizam o ambiente de modo semelhante, porém num espaço de tempo muito maior, resolvendo qualquer dos problemas com a data-limite ainda não vencida. Um aspecto interessante nesse segundo cenário é que ele torna desnecessário qualquer procedimento adicional para envio ou "submissão" da solução. A última versão da solução que tenha sido submetida a teste, é automaticamente registrada em nome do usuário.

Ao solicitar a análise de soluções, o professor pode visualizar, para cada estudante, os problemas trabalhados por ele e, para cada problema selecionado, pode acessar as opções usualmente disponíveis em um ambiente de controle de versões. Dentre essas, o histórico das modificações num mesmo problema, o *download* de qualquer das versões produzidas e as diferenças entre duas versões específicas, são as principais.

5. Discussão e Considerações Finais

Por envolver a manipulação de objetos não tangíveis, a aplicação do método clínico piagetiano em um estudo-piloto sobre aprendizagem de programação teve de ser adaptada. Se por um lado os resultados experimentais foram limitados, por outro eles

possibilitaram a elicitação de requisitos para o desenvolvimento de uma ferramenta de apoio ao registro, organização e análise de programas desenvolvidos por estudantes em disciplinas introdutórias de programação.

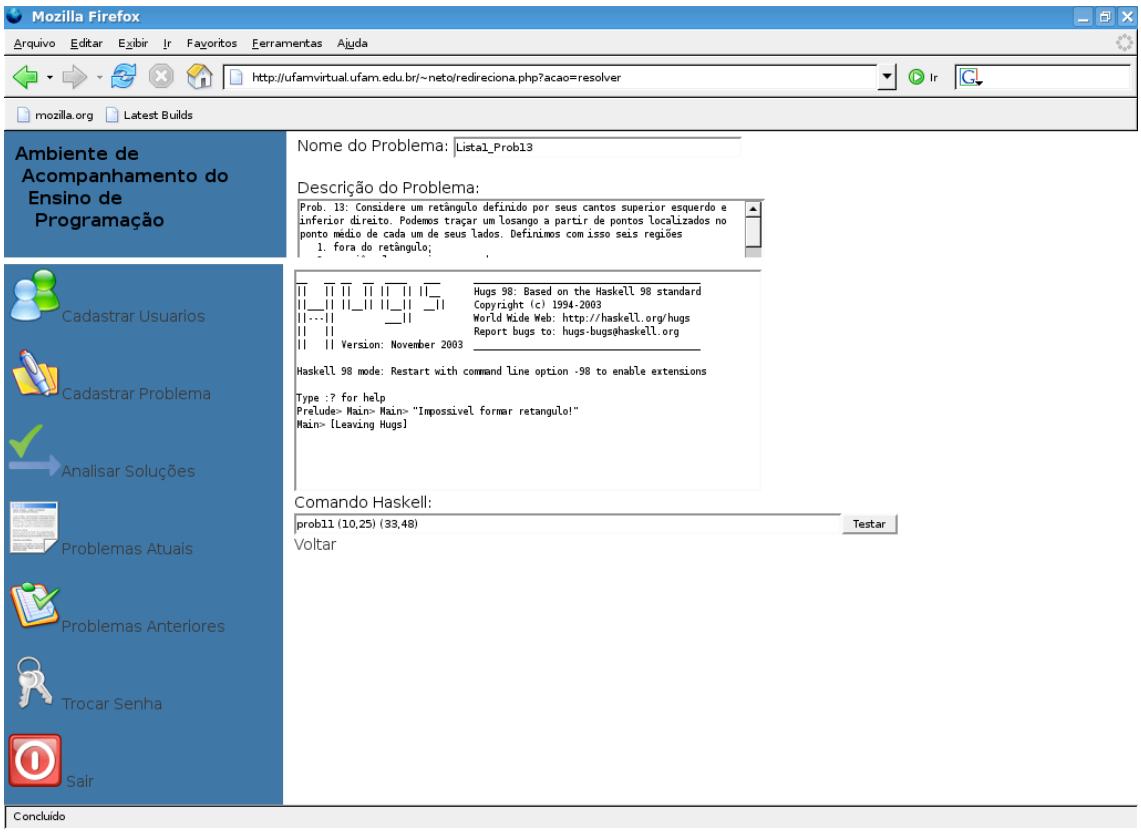

Figura 4 – Testando uma solução no AAEP

Uma vez que é possível visualizar o histórico no desenvolvimento de cada solução do estudante, inclusive comparando diferentes versões dos artefatos construídos por ele, esperamos que a segunda etapa do estudo, ora em andamento, apresente contribuições nos seguintes aspectos:

1) O método clínico poderá ser usado mais efetivamente. As versões intermediárias das soluções poderão ser usadas para explorar nas interações com o estudante, como seus modelos mentais estão ou poderiam estar sendo construídos.

2) Com respeito à característica de saltos nas etapas de planejamento, será possível verificar a consistência entre os resultados obtidos com as observações externas, registros dos próprios sujeitos e entrevistas e a análise do histórico de desenvolvimento das soluções.

3) Espera-se também que a análise do histórico de desenvolvimento de soluções forneça pistas sobre o surgimento de dificuldades em conceitos relevantes, trabalhados em problemas específicos.

4) Também a partir da análise das estruturas intermediárias da solução, espera-se analisar as mesmas segundo os níveis de abstração [castro et al, 2005]

trabalhados: ausência de registro; destaque; nomeação; parametrização e generalização.

Além do registro durante as sessões supervisionadas no laboratório, a ferramenta desenvolvida também está sendo utilizada com sucesso no desenvolvimento de trabalhos não supervisionados que abrangem todos os alunos das três turmas do semestre corrente. Um efeito direto disso é a formação de um repositório de desenvolvimento de programas que pode, num futuro próximo, ser trabalhado por ferramentas semi-automáticas de identificação e feedback em dificuldades na aprendizagem de programação.

Agradecimentos

Este trabalho utilizou recursos do MCT/CNPq, edital CT-Amazônia n.27/2005 (Projeto ColabWeb – Proc. 553329/2005-7).

Referências

Castro, Thais; Castro Jr, Alberto e Menezes, Crediné (2004). "Aprende – um Ambiente Cooperativo de Apoio à Aprendizagem de Programação". Nos anais do XV Simpósio Brasileiro de Informática na Educação. pp. 71-79. Editora EDUA. Manaus, Brasil.

Castro, Thais H.C.; Castro Jr, Alberto N.; Oliveira, Rosane S.C.; Boeres, Maria C.S.; Menezes, Crediné S. (2005) "Enhancing Programming Understanding through Conceptual Schemas in Introductory Courses". CLEI Electronic Journal. Vol. 8, Num. 2, Pap. 4. (http://www.clei.cl/cleiej/).

Delval, Juan (2002). "Introdução à Prática do Método Clínico: descobrindo o pensamento das crianças". Editora ARTMED. Porto Alegre, Brasil.

Engel, G. e Roberts, E. (eds) (2001). "Final Report on Computing Curricula 2001 – Computer Science". In: ACM Journal of Educational Resources in Computing, Vol. 1, No. 3. Dezembro.

Piaget, Jean (1978) "Fazer e Compreender". Edições Melhoramentos. Editora da Universidade de São Paulo, Brasil.

Piaget, Jean e Inhelder, Bärbel (2003). "A psicologia da Criança". Editora DIFEL. Rio de Janeiro, Brasil.