REPRESENTAÇÕES COMPUTACIONAIS AUXILIARES AO ENTENDIMENTO DE CONCEITOS DE PROGRAMAÇÃO

Heloísa Vieira da Rocha*

INTRODUÇÃO

No aprendizado de programação existem conceitos difíceis de serem assimilados, como iteração, recursão, passagem de parametros, etc..

Reconhecendo-se esta dificuldade faz-se uma análise de como é feito o ensino convencional de programação e como as pessoas entendem os conceitos de programação. Desta análise pode-se concluir que o entendimento é falho, mesmo quando se considera pessoas especialistas em computação.

Usando-se como escopo teórico a teoria sobre modelos mentais, foi desenvolvido um sistema de representações computacionais dinâmicas objetivando facilitar o entendimento dos principais conceitos envolvidos no aprendizado de programação.

Neste trabalho é descrita a problemática do ensino convencional de programação; a teoria sobre modelos mentais e a sua relação com o domínio de programação; são apresentadas as características básicas do sistema implementado e os principais resultados do uso experimental deste sistema.

O ENSINO CONVENCIONAL DE PROGRAMAÇÃO

O ensino de programação tem sido uma área muito bem estabelecida, determinado por um paradigma denominado **programação estruturada**, aliado ao uso de linguagens imperativas ou procedurais. De acordo com este paradigma o desenvolvimento de um programa segue um planejamento dirigido do problema para a produção de um programa, estabelecendo uma estrutura fechada de desenvolvimento que compreende os seguintes passos:

Especificar: elaborar um plano detalhado, ou algoritmo, que deverá ser implementado.

Codificar: mapear o plano obtido dentro de construções de uma linguagem de programação.

*Departamento de Ciência da Computação - IMECC - UNICAMP
Núcleo de Informática Aplicada à Educação - NIED

Depurar: corrigir o programa obtido.

Os cursos de programação enfatizam a necessidade de seguir esta sequência de passos para obter um programa. A abordagem está também amarrada às linguagens de programação utilizadas que forçam a estruturação do programa como, por exemplo, a linguagem Pascal.

O passo principal, nesta abordagem, é a elaboração do algoritmo, e nos cursos de uma maneira geral, é passada a idéia de que **aprender a programar significa aprender a elaborar algoritmos.**

Dentro do âmbito da universidade, têm sido oferecidos cursos introdutórios de programação a praticamente todas as carreiras. Existe uma evidente necessidade e interesse em computação. Entretanto os cursos introdutórios de programação, oferecidos para alunos das áreas de exatas e humanas, iniciam suas turmas com uma média de 60 alunos e em poucos meses estão com 20. Obtem-se portanto uma média de 60% de desistência, sugerindo pouco interesse por esta disciplina, o que sem dúvida é uma contradição.

Estamos tendo um fracasso no ensino de programação. Para as pessoas que não questionam a maneira de ensinar, resta a conclusão mais confortável de que infelizmente **programação não é para todos**. Mas, para uma razoável comunidade científica, surgem questionamentos sobre como ensinar programação. São pesquisadores preocupados em entender o processo de aprender a programar, detectando falhas e dificuldades deste aprendizado e sugerindo alternativas, de modo a facilitar o aprendizado de programação.

Um dos principais questionamentos é sobre o paradigma da programação estruturada. Será que ele não está sendo uma "camisa de força", criando obstáculos ao aprendizado de programação? Como consequência, será que as linguagens utilizadas nos cursos iniciais são as mais adequadas? Não se questiona a validade da programação estruturada, mas sim se ela é a melhor maneira de aprender a programar.

Existem linguagens de programação que possuem características projetadas para facilitar o aprendizado de programação.

Um exemplo é o subconjunto gráfico da linguagem Logo que controla a Tartaruga, onde através de um pequeno número de ações bem definidas e claras, resultados interessantes são obtidos rapidamente por principiantes escrevendo programas simples. Com um conjunto de dez instruções mais a estrutura de procedimentos, é possivel um ambiente onde efeitos gratificantes são obtidos nos primeiros contatos com o computador e neste ambiente **todas as pessoas programam**.

Mas, seria ingênuo afirmar que para ensinar/aprender programação basta uma linguagem facilitadora, pois muitos conceitos precisam ser amplamente entendidos para poderem ser utilizados corretamente. Por exemplo, do trabalho com Logo observa-se que as pessoas utilizam procedimentos com parâmetros e não conseguem trabalhar com variáveis dentro de uma atribuição. Apesar de utilizarem variáveis como parâmetros, não possuem a conceituação do que

é uma variável numa linguagem de programação, para que possam generalizar sua utilização. Quanto à estrutura de procedimento e sub-procedimento, verifica-se que as pessoas utilizam a estrutura mas perdem-se ao tentar acompanhar o fluxo de execução de um programa. Fica claro que a simples utilização não implica no entendimento do conceito. Se não for observado este aspecto o quanto antes, não se obtem progresso em programação de modo a desenvolver atividades mais sofisticadas que envolvam amplamente estes conceitos computacionais.

As falhas de conceituação, no trabalho com Logo, são sentidas ao se avançar da parte gráfica para a parte de manipulação simbólica. Ocorrem sérias barreiras, pois o nível de abstração e a complexidade das ações crescem consideravelmente. Além disso, muda-se o paradigma de programação, do procedural para o funcional. Não é mais possível acompanhar a execução do programa o que de alguma forma é conseguido na parte gráfica, pois através da execução de programas que fazem desenhos fica espelhado o comportamento do programa passo-a-passo. Já no processamento simbólico, o que se obtem é simplesmente o resultado, na forma de listas, sentenças, palavras ou números. Torna-se difícil entender como o computador chegou ao resultado, ou não, pois não existem mais maneiras diretas de acompanhar a execução do programa.

Qual a dificuldade no entendimento destes conceitos de programação?

Estes conceitos estão diretamente relacionados ao funcionamento do computador, que no caso é definido pela linguagem, e portanto não é visível. Neste aspecto o computador é uma máquina extremamente opaca, pois se abrirmos um computador e observarmos o que acontece quando um programa está sendo executado, não vamos ver absolutamente nada. A maneira que temos de tentar explicar é através de representações, que de alguma forma espelhem o funcionamento. Permitindo-se esta "visibilidade" do processo de execução, o usuário obtem um melhor "feedback" de sua interação, o que poderá auxiliar no processo evolutivo de construção de seu modelo mental do funcionamento do computador.

MODELOS MENTAIS

" O saber de Jack sobre A nada mais é além de modelos mentais .... que Jack pode utilizar para responder perguntas sobre A" (Minsky, 1985, pg.303)

As pessoas quando interagem com o ambiente, com outras pessoas e com artefatos da tecnologia formam um modelo mental de si próprias, das pessoas e das coisas com as quais estão interagindo. Estes modelos provêem poder de entendimento, capacitando as pessoas a predizer e explicar aspectos da interação. Ao mesmo tempo estes modelos são modificados pela própria interação.

Para principiantes em programação o computador não existe a nível de sistemas binários, portas, processador central e outras unidades básicas. O computador é uma máquina definida em função das construções da linguagem de programação que estão aprendendo. Neste sentido, podemos afirmar que se o principiante inicia programação através da linguagem Logo, o computador para

ele vai ser uma máquina Logo, ou seja, que funciona de acordo com as instruções disponíveis na linguagem Logo, o mesmo irá acontecer com Pascal, ou qualquer outra linguagem de programação utilizada.

Daí definir-se que simplicidade e visibilidade são duas características importantes de uma linguagem de programação para principiantes, pois geralmente eles começam programação com nenhuma idéia sobre as propriedades da *máquina notacional* definida pela linguagem que estão aprendendo. A *máquina notacional* é o modelo idealizado do computador, definido através das construções da linguagem de programação, sendo portanto um computador conceitual idealizado. Para o principiante as propriedades desta máquina são dependentes da linguagem e não do "hardware".

Esta simplicidade e visibilidade facilitam a construção de um modelo mental do computador e da sua forma de operação, este modelo é que irá guiar toda a interação do usuário a nível de programação.

Partindo desta análise advoga-se que instrução deve explicitamente ajudar o estudante a construir um modelo mental de como o computador funciona.

A importância de modelos mentais na aquisição de habilidades humanas é atualmente um tópico de pesquisa bem representado em Ciência Cognitiva e Inteligência Artificial. Na confluência destas duas áreas tem sido desenvolvidas pesquisas que buscam caracterizar, através de um exame cuidadoso, o modo de entendimento das pessoas nos mais variados domínios de conhecimento, como circuitos elétricos (Kleer, 1983), cálculo matemático, física elementar, temperatura e calor, etc.. (Gentner, 1983)

Os resultados destas pesquisas vem sendo utilizados na construção de sistemas especialistas, na concepção de sistemas computacionais realmente amigáveis, no desenvolvimento de material instrucional, manuais, etc..

Também em programação, as pessoas são guiados por modelos mentais de como o código do programa controla as operações do computador. Mas o computador, como sistema físico é extremamente obscuro e fornece um "feedback" muito pobre no sentido de auxiliar um principiante a entender seu funcionamento e consequentemente formar um modelo mental adequado, que o possibilite interagir corretamente.Alguns aspectos do processo de aprender programação em Logo auxiliam a deixar mais clara esta idéia.

O sub-ambiente gráfico é tradicionalmente a "porta de entrada" do Logo. Através da execução de programas que fazem desenhos, de alguma forma se tem espelhado o comportamento passo-a-passo do programa e consequentemente torna-se mais claro o funcionamento da máquina notacional, visualizando o funcionamento de cada construção da linguagem.

Dentro deste ambiente existem diversas componentes que facilitam o usuário a formar um modelo mental do funcionamento do computador.

O computador é visto como um traçador de gráficos que se comporta de forma semelhante a como uma pessoa traça gráficos. Muitas vezes, é solicitado que a pessoa se coloque no lugar da Tartaruga e quase sempre é isto que ela faz quando depura seus procedimentos.

Muitos objetos (conjunto de instruções, Tartaruga, parâmetros) e comportamentos importantes são incorporados ao modelo mental que a pessoa está desenvolvendo sobre o funcionamento do computador, como a sequencialidade na execução de instruções, o procedimento produzindo um resultado visual na tela, parâmetros para generalizar soluções, etc..

O modelo da sequencialidade de execução de instruções guia as pessoas a descreverem o fluxo de execução de um procedimento do tipo:

```
ap casa              ap paredes          ap chaminé
paredes              ......              ......
telhado              fim                 fim
chaminé
janela
fim                  ap telhado          ap janela
                     .....               ......
                     fim                 fim
```

como sendo, **casa -->paredes --> telhado ---> chaminé ---> janela**

Como o "feedback" que é dado pelo computador não explicita que este não é o fluxo, as pessoas persistem com esta idéia de fluxo e com este modelo de funcionamento.

Sem muita dificuldade é incorporado o comportamento iterativo, por ser uma idéia decorrente do sequencial e que somente simplifica a escrita de comandos. O mesmo não acontece com comandos condicionais. Dificilmente se observa principiantes escrevendo comandos condicionais com a estrutura se/então/senão, que implica em uma quebra maior da sequencialidade estrita, pois dependendo do resultado da expressão utilizada no comando, algumas instruções deverão ser "saltadas".

O uso do modelo iterativo, de repetição, guia o entendimento de procedimentos recursivos. Isto pode ser observado, na escrita de procedimentos do seguinte tipo:

```
ap contar :n
esc :n
se :n = 0 [ pare ]
contar :n-1
esc :n+1
esc :n+2
esc :n+3
fim
```

para escrever a seqüência de números 3 2 1 0 1 2 3.

Através deste exemplo extraído de (Valente, 1988), está claro que recursão é vista como uma repetição. Também pode ser observada a necessidade em controlar o processo concretamente, que é abstraído do uso inicial da linguagem no modo direto. E mais uma vez a resposta que o computador vai dar a este procedimento não irá auxiliá-lo a formular o correto modelo de

recursão que é o de cópia (a cada chamada recursiva uma nova cópia do procedimento é ativada), pois ele não consegue visualizar o processo de execução.

Estes problemas são salientados, no aprendizado de Logo, quando se tenta avançar da parte gráfica para a parte de manipulação simbólica. O modelo do computador não é mais o de um traçador de gráficos, precisa ser modificado sensivelmente.

Esta mudança é agravada pois Logo perde a característica de visibilidade. Não se pode mais acompanhar a execução do programa. O que é obtido é simplesmente o resultado, na forma de listas, sentenças, palavras ou números. Torna-se difícil entender como o computador chegou ao resultado, ou não, pois não existem mais maneiras diretas de acompanhar a execução do programa. Além disso, muda-se o paradigma de programação, ou seja, agora deve ser reformulado também o modelo de procedimento, acrescentando o comportamento funcional. De acordo com a teoria de Williams em (Williams, 1983) as regras de comportamento associadas ao objeto autonômo procedimento devem ser substituídas, de forma a acomodar a definição de procedimento que é uma operação, ou então, um novo objeto denominado operação deve ser incorporado ao modelo.

Norman em (Norman, 1983) ressalta que ao se analisar modelos mentais, com relação a sistemas físicos, é necessário considerar quatro componentes:

.**o sistema objetivo**, que por definição é o sistema que a pessoa está aprendendo ou usando.

.**o modelo conceitual do sistema objetivo**, que são ferramentas para auxiliar o entendimento ou ensino de um sistema. São inventados por professores, designers e engenheiros com o objetivo de prover uma representação adequada do sistema objetivo. É desejável que sejam precisos, consistentes e completos.

.**o modelo mental do usuário sobre o sistema objetivo**, é o que as pessoas realmente tem em suas cabeças e que guia o seu uso do sistema.

.**a concepção do cientista sobre a natureza modelo mental**, que é, claramente, um modelo do modelo.

Na definição de um modelo conceitual, sobre um sistema ou processo deve-se levar em conta qual a natureza (Young, 1983) de modelo mental (analogia forte, vocabulário ou substituto) que se está considerando, para que se obtenha a desejável correlação entre o modelo mental e o modelo conceitual.

O tipo de natureza de modelo que fundamentou a concepção do modelo conceitual implementado e descrito neste trabalho foi o substituto ("Surrogate Model").

Um **modelo substituto** M para um sistema S é essencialmente uma noção familiar de um modelo de trabalho o qual explica como S funciona como um mecanismo. Tipicamente M apresenta uma abordagem bastante simplificada de S, de forma a ter "*mais o sabor de uma estória de ficção do que de uma elaborada descrição de engenharia*" (Young, 1983).

M é chamado de substituto porque ele pode ser usado no lugar de S para responder questões sobre o comportamento de S. Em adição a ser amplamente utilizado em Ciências e Engenharia, modelos substitutos tem sido utilizados como um meio de fornecer a programadores novatos um esquema de como funciona o interpretador de uma linguagem de programação (Boulay, 1981; Mayer, 1981).

Modelos substitutos são guiados pela propriedade predectiva dos modelos mentais e, primariamente, podem ser vistos como um simulador do sistema objetivo. Também são de grande auxílio, ao prover uma estrutura para o funcionamento do sistema e como conseqüência auxiliar a pensar sobre ele.

CARACTERÍSTICAS BÁSICAS DO SISTEMA IMPLEMENTADO

O sistema fornece diversas representações da execução de um programa escrito em Logo não gráfico. Os aspectos principais do sistema são a multiplicidade de representações e a característica dinâmica das representações, ou seja, elas são fornecidas em tempo real, ao mesmo tempo em que é executado o procedimento.

O sistema não tem como objetivo ensinar os conceitos e sim oferecer uma alternativa que facilite seu entendimento. É o *fazer ver para melhor entender*.

Foi definido um conjunto de representações baseado nas representações utilizadas estaticamente, adicionando-se primordialmente a característica de tempo real. Os conceitos computacionais considerados no desenvolvimento do sistema foram fluxo de execução, variáveis e recursão. Acredita-se serem estes os conceitos fundamentais de programação, e todos os demais, como iteração, procedimentos, etc., podem ser tratados em função deles.

Existem duas formas básicas de representações denominadas locais e globais.

O subconjunto de representações locais tem como objetivo mostrar num certo ponto de execução o que está acontecendo e o que o computador "conhece" a respeito do programa. Perde-se nestas representações a história passada da execução, ou seja, não é visível o processo que conduziu aquele momento da execução. Por acreditar que visualizar de alguma forma o processo como um todo é importante para o entendimento dos conceitos abordados, principalmente com relação aos aspectos de chamada e retorno de procedimentos e programas com muitos níveis de execução, definiu-se outra representação que tenta mostrar estes aspectos da história de execução do programa. Esta outra representação denominou-se de global.

Quanto à representação local, o sistema proporciona uma tela subdividida em três janelas, como pode ser visto na **Figura 1**.

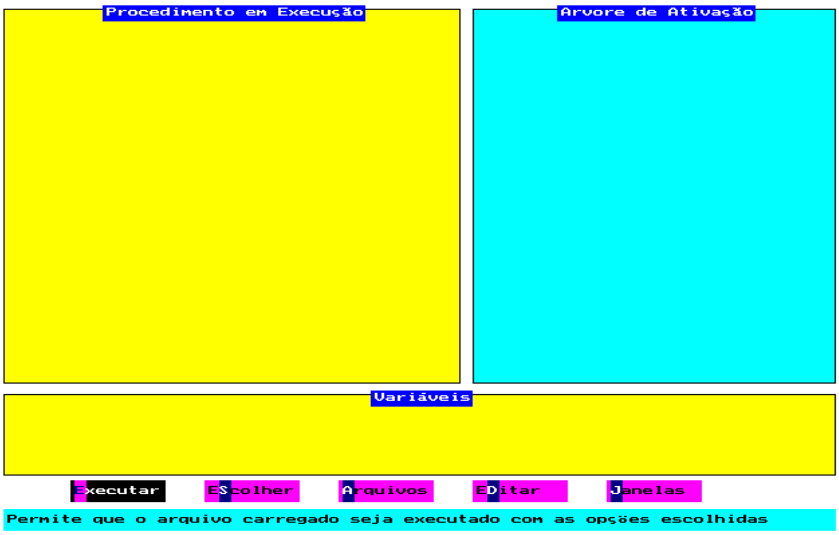

Figura 1

A primeira janela, nomeada **"Procedimento em Execução"**, contém o texto do procedimento em execução e o cursor em vídeo reverso, ou em outra cor se o vídeo for colorido, mostra a instrução que está sendo executada. Através desta representação pretende-se deixar claro, dentre outros aspectos, a sequencialidade de execução das instruções.

No sistema quando ocorre a chamada de um sub-procedimento, o texto do procedimento chamante é substituído pelo do procedimento chamado, e uma mensagem informa que está ocorrendo uma chamada. Isto pode ser observado nas **Figuras 2 e 3**.

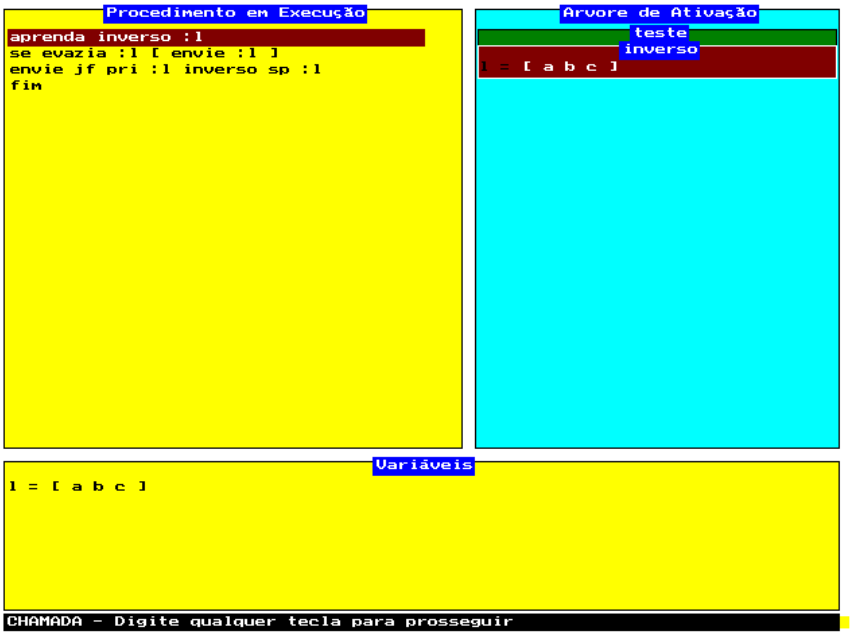

Figura 2

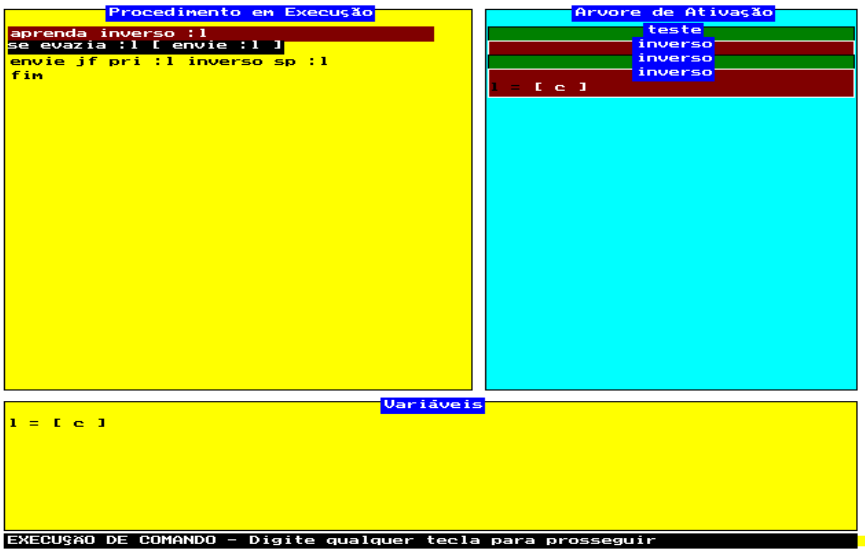

Figura 3

O mesmo ocorre quando do retorno do sub-procedimento, onde uma mensagem informa que está ocorrendo um retorno, e volta à tela o texto do procedimento chamante e a execução prossegue. O mesmo esquema é utilizado no caso de procedimentos recursivos. Obtem-se então uma representação de como é efetuada a chamada e retorno de procedimentos.

Na segunda janela, nomeada **"Arvore de Ativação"**, é representada a estrutura dinâmica de chamadas de procedimentos. Nesta janela para cada chamada é criado um retângulo com nome do procedimento chamado e que contém o nome e valor dos parâmetros no momento da chamada. Estes retângulos vão sendo construídos e eliminados de acordo com a ocorrência de chamada ou retorno da chamada de procedimentos. (Figura 2 e 3)

A terceira janela, nomeada "**Variáveis**", contém o nome das variáveis ativas e de seus valores correntes. Cada alteração do valor de uma variável dentro do programa é refletida nesta janela com realce. O objetivo desta terceira janela é ressaltar os aspectos relativos à conceituação de variáveis e seu escopo de validade, mostrando variáveis que "passam a existir" no momento da chamada de um procedimento e que "deixam de existir" no retorno.

Pode ser observado (Figura 1) que além das janelas, existe um menu de opções que irá coordenar a execução do sistema. Através desse menu pode-se definir qual o modo de execução (passo-a-passo ou direto), qual o arquivo a ser carregado, qual o procedimento com o qual se deseja iniciar a execução, qual o tamanho das janelas a serem vistas, etc..

Existe uma quarta janela, como pode ser visto na **Figura 4**, nem sempre presente na tela, onde são pedidos valores de entrada e onde são mostrados valores de saída. Esta janela somente é aberta quando da execução de comandos de entrada e saída e desaparece ao término da execução dos comandos respectivos, ou quando o usuário desejar.

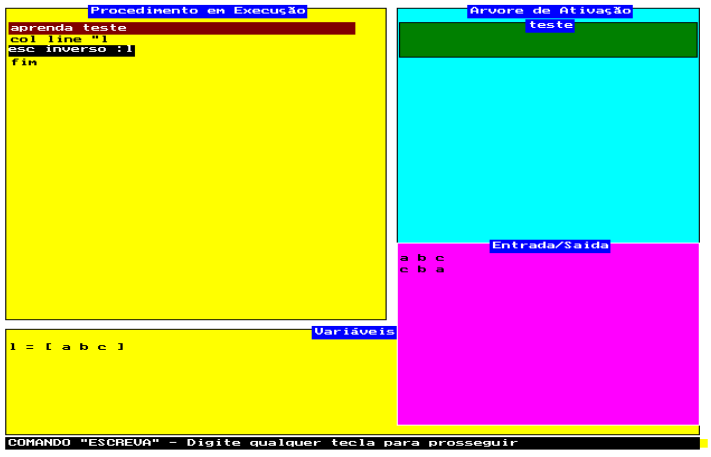

Figura 4

Outra janela, que também não está sempre presente na tela, indica saídas relativas ao comando ENVIE (retorno de função). Como pode ser visto na **Figura 5** nesta janela tem-se o valor que está sendo enviado e os respectivos procedimentos de origem e destino. Também esta janela somente aparece quando do retorno de uma operação e desaparece no momento em que é concluído o retorno ou sob comando do usuário.

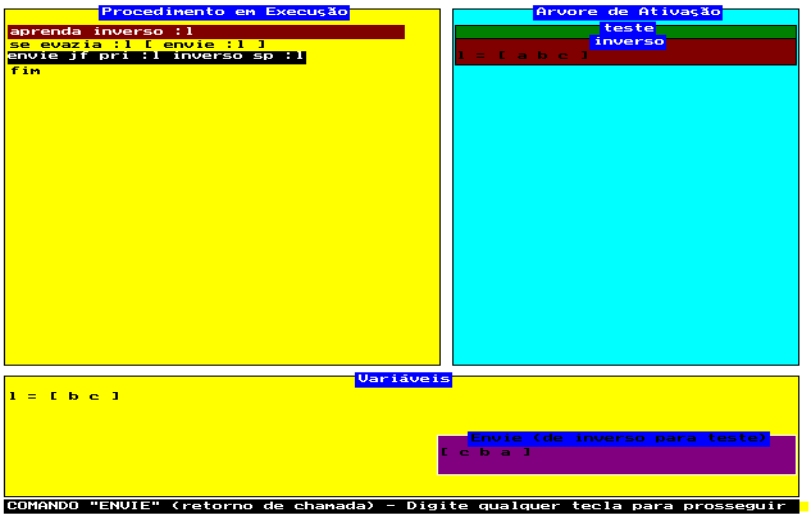

Figura 5

Através de uma tecla de função pode-se ter acesso à representação global da execução. Esta representação é feita em tela única e mostra o esquema de execução dos procedimentos. Nesta tela tem-se o texto de todos os procedimentos que compõem a área de trabalho, ou seja, o arquivo utilizado na corrente execução. Através de um esquema de cores, são utilizados cursores que indicam qual a instrução que está sendo executada e quais as execuções em suspenso, ou seja, pontos de chamada e retorno nos outros procedimentos ativos da área de trabalho.

Toda interação com o sistema é feita através da seleção de opções em menu. A qualquer momento da execução de um programa pode ser interrompida e se ter acesso a outro tipo de representação ou outro modo de execução.

RESULTADOS DO USO

O sistema foi utilizado experimentalmente (Rocha, 1991) em três situações e com os seguintes objetivos:

. diagnosticar e remediar o conceito de recursão

. aprendizado de recursão no processamento simbólico

. verificar o quanto especialistas em computação dominam o conceito de recursão

O primeiro uso do sistema, foi feito durante uma oficina de trabalho que tinha como um dos objetivos oferecer um aprofundamento em programação Logo. Os participantes eram um grupo de pesquisadores e técnicos em Informática e Educação que conheciam e estavam desenvovendo atividades de programação com a linguagem Logo.

O segundo uso foi mais prolongado e foi feito em um curso de 40 horas, oferecido para um grupo de professores da rede pública de ensino participantes de um projeto de introdução de computadores em escolas, utilizando programação Logo. Apesar da necessidade de um aprofundamento para poder conduzir o projeto que estavam envolvidos, achavam a parte de manipulação simbólica muito difícil e muito diferente do Logo que conheciam. Dentro desta perspectiva tinha-se como ponto central de observação o quanto o sistema iria viabilizar este aprofundamento junto a este tipo de usuário, que necessitava de programação mas não podia ser qualificado de especialista.

O terceiro uso experimental foi feito por um aluno com formação em computação e que estava fazendo pós-graduação em computação. Ele conhecia Logo bastante bem e portanto se mostrou bastante adequado para que se pudesse desenvolver a observação que se tinha interesse. A idéia de trabalhar com este aluno foi uma consequência de uma atividade feita com alunos do curso de Ciência da Computação. Através desta atividade pode-se constatar que mesmo alunos de um curso específico de computação apresentavam sérias falhas no entendimento de um processo recursivo e estas falhas não estavam sendo percebidas, dentro dos moldes como eram efetuadas avaliações.

Deste trabalho experimental pode-se concluir que podemos representar o modelo mental que os estudantes possuem de programação através do seguinte esquema:

Com os materiais instrucionais tradicionalmente utilizados e o "feedback" que é obtido pelo ambiente de programação, os estudantes são capazes de desenvolver um modelo mental de base

limitado e falho em muitos aspectos. Todos os programas cujo comportamento não consegue ser explicado através deste modelo de base são "agregados" como padrões.

Como o modelo de base não sofre alteração (evolução), somente são desenvolvidos e entendidos os procedimentos que possam ser explicados e os que "encaixam" em algum padrão. O modelo de base para recursão, quando existia, incluia recursão como repetição, e os alunos possuiam padrões de procedimentos recursivos, com finalidades específicas, que envolviam retorno. Caso fosse solicitado um procedimento, que envolvia retorno de recursão e que fosse de natureza diversa da dos padrões agregados, não havia entendimento. Para exemplificar, pode-se mencionar um protocolo descrito em (Rocha, 1992) onde o aluno de computação naturalmente resolvia sofisticados problemas de percurso em estruturas de dados como árvores, mas não conseguia identificar como correto um procedimento bastante simples de atualização de campos de uma lista, quando esta atualização era feita no retorno da recursão.

O que está sendo perdido neste processo de ensinar através de uma enorme quantidade de exemplos, é focalizar à atenção nos conceitos básicos e que não são muitos: variável, iteração, procedimento/operação, fluxo de execução e recursão. Além disso, ao utilizar o computador como auxílio, não se pode perder que a aquisição de conceitos envolve um ciclo de fazer, refletir sobre o que fez, depurar, fazer, refletir,.... O computador deve ser inserido neste processo de aprendizado favorecendo, e não ensinando, que as pessoas possam individualmente através da exploração formar seu próprio modelo do conceito. Afinal, nunca nos foi ensinado sobre gatos e sobre o que os diferencia de cachorros. É preciso deixar o objeto programação passível de ser manipulado, de modo que o aprendiz possa interagir com esses objetos computacionais do mesmo modo que interage com animais e é capaz de aprender a diferença entre gato e cachorro.

É preciso dar visibilidade ao processo de execução e isto deve ser feito por quem efetivamente executa o programa, ou seja, o computador. E a visibilidade deve ser dada ao programa que o aluno fez. O processo de fornecer representações diferentes das usuais deve levar em conta que o computador possui características que o qualificam como a melhor mídia para provocar estas mudanças. As representações devem ser dinâmicas, como o processo que estão tentando espelhar. O ambiente deve ser interativo. Não interessa fornecer a representação de um procedimento padrão e não permitir a interação do usuário, na definição e visualização de seus próprios procedimentos. Isto é que torna diferente e válido o uso do computador como ferramenta auxiliar ao ensino, destacando-o sensivelmente de outras mídias como o vídeo na produção de um filme, ou de um livro em sua forma estática. E foram estas características que basearam o sistema descrito neste trabalho.

Utilizou-se o computador por achar que as formas de representação tradicionais como os diagramas de execução, brincar de computador, árvores de ativação, etc., precisavam ser mudadas na sua essência. Do estático deveria-se passar para o dinâmico, pois o processo de execução de um programa é dinâmico. Do mostrado em um papel ou feito por um professor, deveria ser feito pelo próprio sistema que executa o processo, o computador. E a representação ou visibilidade deveria ser dada ao programa feito pelo aluno e não para um exemplo padrão feito por um perito. E o usuário deveria ter meios de interferir no processo, controlando a forma como gostaria de observar e para quais exemplos.

Os resultados do uso confirmaram as hipóteses iniciais de que a visibilidade dada através de sistemas dinâmicos, realmente instrumenta as pessoas a pensarem sobre programação. Algumas conclusões devem ser ressaltadas:

. **Entender como funcionam as construções de uma linguagem não é suficiente para entender programação.** A maioria das pessoas observadas não conseguia a princípio sintetizar o funcionamento de um procedimento. Sempre que eram solicitadas a dizer o que um procedimento fazia, efetuavam uma leitura comando a comando, explicando-os isoladamente. Para entender programação é preciso entender que quando agrupados, na forma de um programa ou de um procedimento, os comandos deixam de ter um significado e estrutura descontextualizada. Este aspecto deve ser enfatizado no ensino de programação, pois sem isto nunca o aluno irá conseguir adquirir estratégias de solução de problemas como a de dividir um problema em partes. Portanto, não adianta efetuar um extensivo trabalho na área de solução de problemas se o entendimento do processo de execução está falho. O ambiente de programação pode auxiliar neste aspecto tornando o processo mais explícito, nos moldes do sistema desenvolvido.

. **Programar sob o paradigma procedural é a melhor maneira de iniciar programação**. Isto ocorre por duas razões: primeiro, por ser este o paradigma mais próximo do real funcionamento do computador, e por consequência leva o aluno a criar um modelo mental mais adequado de como o computador funciona; segundo, por ser este o paradigma mais próximo de como as pessoas atuam no dia-a-dia. É muito eficaz a metáfora de ensinar uma nova palavra ou um novo comando ao computador, explicando o seu significado, e que uma vez ensinado pode ser utilizado sem precisar redefinir. Tendo este aspecto deve-se levar em conta que mudar um paradigma de programação envolve muito mais que mudar a estratégia de resolver um problema. Irá envolver a mudança do modelo mental que a pessoa faz do comportamento do computador. Isto deverá ser reformulado para que a pessoa possa controlar a máquina efetivamente sob o novo paradigma. Portanto, toda mudança de paradigma também deverá envolver a invenção de um ambiente de representações que favoreça este novo entendimento da máquina notacional que está sendo desenvolvida.

. O entendimento de recursão, fluxo de execução, variáveis globais e locais, pilha de execução, etc.. está profundamente ligado ao entendimento que as pessoas têm de como o processo é efetivamente executado pelo computador. E para que este entendimento ocorra é absolutamente necessário que se dê oportunidade de visualização do processo. Um exemplo claro é o trabalho efetuado com o aluno de computação. Este aluno era sem dúvida um bom programador, mas não tinha elaborado completamente o conceito de recursão, pois a recursão com volta nunca era utilizada. Escrevia muitos procedimentos padrões que envolviam volta, mas não conseguia explicar seu funcionamento, e quando a proposta era de um procedimento novo nunca visto, ele resolvia o problema mas nunca utilizava o retorno da recursão. O processo com que ele utilizou o sistema não foi para auxiliar no desenvolvimento de novos programas, mas sim para visualizar o processo de execução de programas que ele já conhecia. E com isto ele reformulou muitas das soluções já apresentadas, achando que utilizar a recursão com volta deixava o "programa mais claro". Portanto tornar o processo explícito auxilia, e evidentemente o grau de auxílio é proporcional ao tipo de estrutura mental que o usuário dispõe em decorrência de sua experiência prévia com programação.

Portanto é preciso pensar em alternativas ao usual. Novas ferramentas devem ser adicionadas ao ensino, levando em conta os processos cognitivos envolvidos no aprendizado de programação. Quanto ao nível de entendimento da máquina notacional, acredita-se que o esquema de representações computacionais dinâmicas que espelhem o funcionamento da máquina através de um modelo conceitual coerente com o modelo mental que queremos seja construído pelo aluno, é uma boa alternativa.

CONCLUSÕES

De modo geral, mais pesquisa precisa ser feita para determinar o efeito específico de modelos no que está sendo aprendido, e para determinar as características de um modelo conceitual adequado.

Modelos conceituais são ferramentas para o entendimento ou ensino de sistemas. São inventados para prover uma representação adequada do sistema em uso ou que está sendo ensinado, consistindo a base do modelo mental que se deseja que o usuário forme do sistema. Idealmente, quando um sistema é construído, seu projeto deveria ser baseado no modelo conceitual.

O modelo conceitual deveria governar toda a interface humana com o sistema e dessa maneira a imagem do sistema que é vista pelo usuário seria consistente, coesa e inteligível. O que geralmente se tem é uma imagem do sistema distante do modelo conceitual.

Portanto, se instrutores e materiais instrucionais do sistema ensinam ao usuário o modelo conceitual de base, e se a imagem do sistema é consistente com este modelo, o modelo mental do usuário será consistente.

Os resultados obtidos demonstraram que existem falhas conceituais sérias no aprendizado de programação e que são difíceis de serem superadas com metodologias tradicionais. Também demonstrou-se que um sistema que torne mais visível o processo de execução de um procedimento pode ser útil na aquisição destes conceitos.

Finalmente, o uso de sistemas computacionais dinâmicos de representações deve ser avaliado em outros domínios de conhecimento que também envolvam conceitos decorrentes do entendimento de processos abstratos. Estudar novas formas de representação que possam ser utilizadas em áreas de Ciência onde conceitos como equilíbrio, pressão, temperatura, etc., são extremamente dependentes do processo onde ocorrem. Outra área de muito interesse e que inclusive originou este tema de pesquisa, é Eletricidade. A idéia é procurar outras formas de representação (cores, som, analogias com outros domínios, etc.), diferentes da tradicional, que auxiliem as pessoas a entenderem conceitos como voltagem, corrente alternada, corrente contínua, potência, etc., precisam ser analisadas.

E em todas estas áreas, deve ser avaliada a importância de modelos mentais, como são criados, qual sua natureza, para se poder construir representações diferentes das usuais, sistemas cognitivamente eficazes e portanto modelos conceituais dinâmicos e úteis.

REFERÊNCIAS BIBLIOGRÁFICAS

Boulay, Benedict Du, O'Shea, Tim e Monk, John (1981), The Black Box Inside the Glass Box: Presenting Computing Concepts to Novices, Int. J. Man-Machine Studies 14, 237-249

Gentner, Dendre e Stevens, Albert L. (eds.) (1983), Mental Models, Lawrence Erlbaum Associates, Publishers

Kleer, Johan de e Brown, John Seely (1983), Assumptions and Ambiguities in Mechanistic Mental Models in Gentner, D. e Stevens, A. L. (eds) *Mental Models* Lawrence Erlbaum Associates, Publishers

Larkin, Jill H. e Simon, Herbert A. (1987), Why a Diagram is (Sometimes) Worth Ten Thousand Words Cognitive Science, 11, pp.65-69

Mayer, Richard E. (1981), The Psychology of How Novices Learn Computer Programming, Computing Surveys, Vol 13, n. 1

Minsky, M. (1985), The Society of Mind, Simon & Shuster, NY

Norman, Donald A. (1983), Some Observations in Mental Models, in Gentner, D. e Stevens, A. L. (eds) *Mental Models* Lawrence Erlbaum Associates, Publishers

Rocha, H.V. (1991) Representões Computacionais Auxiliares ao Entendimento de Conceitos de Programação, tese de doutorado, FEC, UNICAMP

Valente, Ann B. (1988), Aprender Programação Logo: Será que algumas pessoas tem mais facilidade que outras? Núcleo de Informática Aplicada à Educação, Memo 16, UNICAMP