Núcleo de Informática Aplicada à Educação
Universidade Estadual de Campinas

Resumo

Os usuários de LOGO, mais freqüentes têm sido a criança, o adolescente ou o adulto em sua primeira experiência com computadores. O presente trabalho relata a experiência com LOGO de universitários de uma área afim (ciência de computação). Essa categoria de usuários permitiu avaliar aspectos de LOGO que não emergiam nas populações usuais. Em especial, projetos relativamente extensos possibilitam uma visão mais ampla dos "estilos de programação" subjacentes.

NIED - Memo N° 11
1988

Estudos sobre Estilos de Programação em LOGO

Maria Cecília Calani Baranauskas

Cidade Universitária "Prof. Zeferino Vaz"
Prédio V da Reitoria - 2º Piso
13083-970 - Campinas - SP
Telefones: (019) 3788-7350 ou 3788-7136
Fac-símile: (19) 3788.7350 e 3788.7136 (ramal 30)

Estudos sobre Estilos de Programação em Logo

Maria Cecília Calani Baranauskas[1]

1 Introdução

A metodologia de construção de programas tem passado por vários estágios ao longo do tempo. A idéia mais antiga parece ter sido a de escrever programas para uma máquina específica onde ele deveria ser executado. Durante a década de 70, a idéia de programação estruturada revolucionou o conceito de programação, colocando outros pontos relevantes durante a construção de programas. Passaram a fazer parte da metodologia aspectos importantes como:

- especificação do projeto, minimizado a distância entre a estratégia adotada para a solução do problema e o texto do programa.
- análise sob o aspecto correção do programa.
- manutenção e possibilidade de expansão considerando seu uso por pessoas não envolvidas em sua confecção.

A partir de então, ao lado da utilização dos recursos de determinada linguagem, clareza de expressão tem sido fortemente encorajada.

A extrapolação dessa idéia para sistemas grandes, onde a visão do todo deveria ser igualmente clara, levou ao desenvolvimento de técnicas de documentação e motivou o aparecimento de linguagens para especificação e interconecção de módulos, (MODULA2, por ex.) apropriada para grandes sistemas.

Uma nova abordagem está sendo proposta para a próxima década por Knuth (1984), chamada "literate progamming". A idéia de Knuth é mudar a atitude tradicional na construção de programas, segundo palavras do próprio autor: *"em vez de imaginarmos que nossa tarefa é instruir o computador a fazer algo, devemos nos concentrar em explicar a seres humanos o que queremos que o computador faça".* Com base nessa filosofia, Knuth desenvolveu um sistema chamado "WE" (Bentley, 1986), que possibilita ao seu autor trabalhar com o código e com o texto (documentação) de seu programa simultaneamente, sem impor um determinado estilo.

[1] Departamento de Ciência da Computação – IMECC
& Núcleo de Informática Aplicada à Educação
Universidade Estadual de Campinas – São Paulo

De certa forma a linguagem LOGO (1985) enfatiza, aspectos presentes na metodologia chamada "programação estruturada", ao mesmo tempo em que parece endossar a filosofia proposta por Knuth, quando propõe que um programa em LOGO deve ser por si só explicativo, inteligível.

Por outro lado, é de fundamental importância, para o "ensino de programação", considerando o estado da arte em metodologia de construção de programas, que se identifique e explore características de estilo presentes na atividade de programar com a forma de aprimorar a habilidade de escrever programas para resolver problemas.

O termo "estilo", definido no dicionário (Holanda, 1985), pode ser usado simplesmente como uma forma de expressão:

- "maneira de exprimir os pensamentos falando ou escrevendo"

ou pode carregar um calor:

- "maneira de escrever correta e elegante".

Definiremos "estilo de programação" como uma maneira de expressar um algoritmo numa linguagem de programação. Um "valor" pode ser associado a determinado estilo dependendo da ótica com que se "olha" o objeto programa.

O presente trabalho pretende fazer uma análise crítica de projetos desenvolvidos em LOGO, por alunos ingressantes na Universidade, durante seu primeiro curso de 'programação de computadores', com respeito a características de estilo de programação.

2 Metodologia

2.1 O Contexto

Foram observadas cerca de 40 alunos durante um período de dois meses, trabalhando com LOGO como parte do conteúdo de uma disciplina introdutória de programação de computadores. Tal população, "calouros do curso de Ciências de Computação" tinham a motivação natural de estarem cursando a primeira disciplina afim de seu curso, isto já no primeiro semestre na Universidade.

O trabalho com LOGO em tal disciplina foi feito em caráter experimental e a idéia era usar da simplicidade do LOGO para introduzir os conceitos básicos envolvidos no aprendizado de linguagens de programação, como por exemplo o conceito de variável, estruturação de programas,

recursão, etc. para em seguida trabalhar com Pascal, linguagem oficial da disciplina. A idéia era se trabalhar com desenvolvimento de algoritmos sem a preocupação com especificidades de linguagem (detalhes como ';' , declarações de variáveis e tipos) e ao mesmo tempo oferecer um ambiente lúdico capaz de suportar aplicações interessantes.

Os alunos participavam de quatro horas/aulas por semana, das quais duas horas eram no computador além de horas livres que dedicavam a trabalhos individuais. Trabalhavam em grupos de dois e, com base na metodologia LOGO, sempre desenvolvendo pequenos projetos propostos pelo professor com objetivo de explorar determinado conceito. Leitura individual de livro-texto (Valente, 1985) antecedia o trabalho no computador.

Como avaliação para esse período inicial, foram propostos alguns "temas" para o desenvolvimento de projetos em LOGO. Entre os temas apresentados estão os seguintes:

- "Faça uma reflexão de sua experiência com LOGO e implemente um 'demonstrativo' do que é LOGO. (Imagine que você o apresentaria numa 'Feira de Informática')".

- _ "Pense em como fazer 'animação' em LOGO. Estruture um projeto para mostrar características da animação. Possibilite interação ao usuário".

- "Imagine utilizar LOGO para 'ensinar' algo específico. Proponha um conteúdo a ser trabalhado, idade aproximada do aprendiz, etc. Projete as ferramentas básicas de que necessitará".

- "Implemente um jogo interessante".

- "Proponha uma maneira de fazer 'LOGO em três dimensões'. Projete as ferramentas básicas necessárias".

2.2 Análise dos Projetos

Foi feita uma análise dos projetos com o objetivo de identificar estilos de programar, tendo em vista os seguintes aspectos: definição do projeto, estruturação, controle de execução e uso dos recursos da linguagem. Em definição foi observado o processo de transformação que vai do entendimento do problema até o algoritmo utilizado para resolvê-lo. Em estruturação foi observado tanto o aspecto 'forma' como aspecto 'conteúdo', em relação ao conjunto dos procedimentos que definem propriamente o projeto. Em controle foram observados maneiras de controlar o fluxo de execução dos procedimentos como um todo bem como o tipo de estruturas de controle encontradas

nos procedimentos individualmente. O aspecto 'uso dos recursos da linguagem' refere-se principalmente à sub utilização de recursos, e sua relação com os itens controle e estruturação.

3 Discussão dos Resultados

3.1 Definição do Projeto

Os projetos em geral são constituídos por conjuntos de procedimentos que trabalham as informações mais um mecanismo de controle do fluxo de execução. Com relação a esse aspecto foram encontrados, basicamente, dois "estilos" para projeto:

- em um deles, que chamaremos "A", o controle está concentrado em determinado(s) procedimento(s); portanto, o projeto como um todo é constituído por procedimentos com dois tipos de funções muito bem definidas: procedimentos "de trabalho" e procedimentos "de controle".

- um segundo estilo, que chamaremos "B", o controle aparece espalhado por vários procedimentos, o que torna a visão do projeto como um todo um tanto "nebulosa".

Exemplo de um projeto no estilo "A":

projeto: "Menu de Desenhos"

descrição dos procedimentos:

proc. desenhar: controla toda a execução do programa.

proc. menu: mostra ao usuário, desenhos disponíveis permitindo a ele escolher um deles.

proc. gravar: permite ao usuário colocar sua composição na lista de desenhos disponíveis.

proc. incluir: permite ao usuário ensinar novos desenhos básicos.

```
ap desenhar
dt
col "" listaproc
col jf [] [] "listaproc
menu
esc [quer gravar a composição? (s/n)]
se care = "S então [gravar]
esc [quer incluir novo desenho básico? (s/n)]
se care = "S então [incluir]
esc [quer continuar a desenhar? (s/n)]
se care = "n então [ gravetudo "desenhar! tchau]
tat
desenhar
fim
```

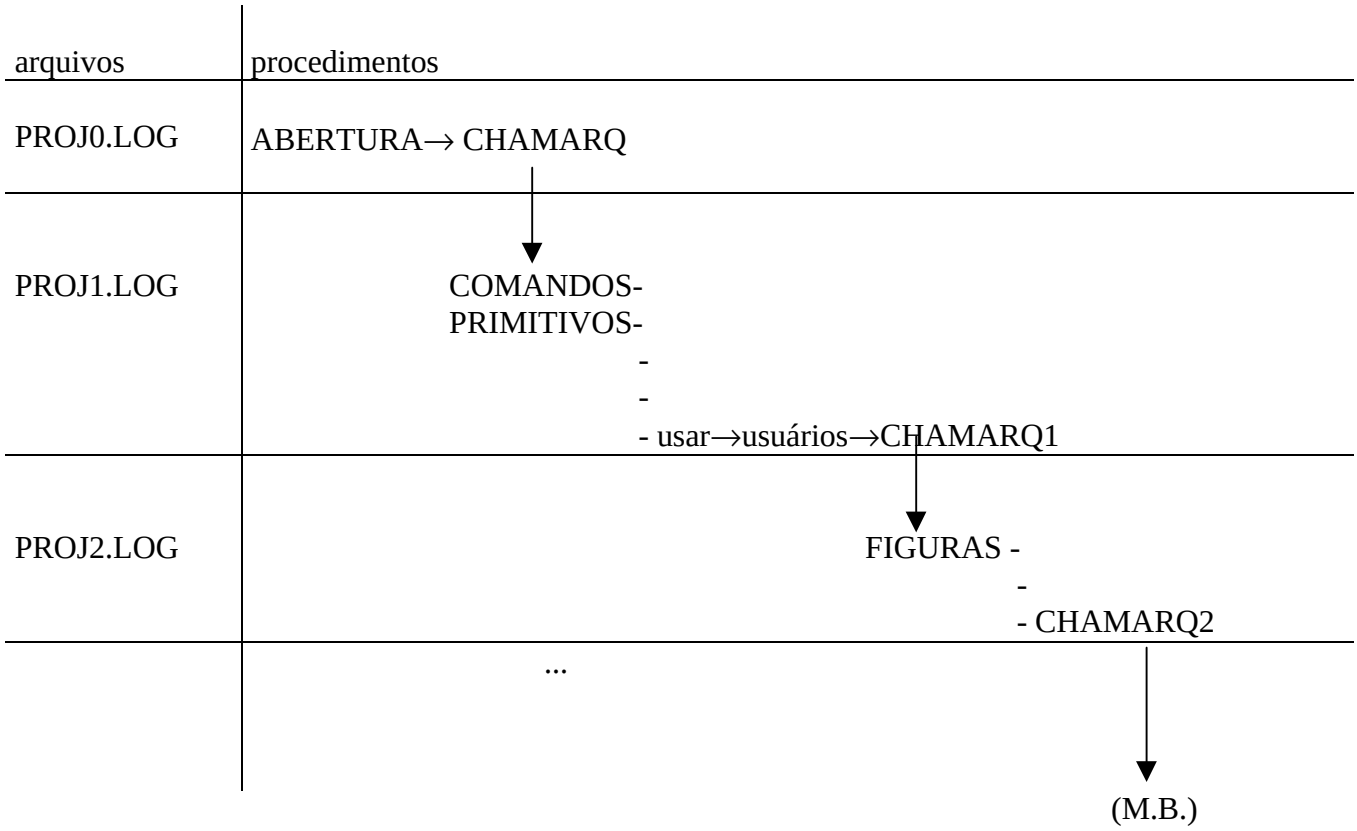
```
ap mar          ap montanha     ap barco        ap nuvem
.               .               .               .
.               .               .               .
.               .               .               .
fim             fim             fim             fim

ap praia        ap bola         ap gravar       ap incluir
.               .               .               .
.               .               .               .
.               .               .               .
fim             fim             fim             fim
```

(W.M.)

Neste projeto o procedimento "desenhar" faz o controle do fluxo de execução dos procedimentos "mar", "montanha", "gravar", etc., que são ferramentas básicas, aqui chamadas procedimentos de "trabalho".

Exemplo de projeto no estilo "B":



"chama"
abertura-------->chamarq--->...--->chamarq1--->...--->chamarq2--->...--->...--->chamarq6--->final

Ou seja, não existem procedimentos que fazem apenas "controle", enquanto que os outros são apenas "ferramentas básicas".

Parece não haver uma relação determinada entre os estilos A e B e qualquer dos métodos clássicos de desenvolvimento de programas "top - down" e botton - up".

3.2 Estruturação

Pode-se analisar o aspecto "estruturação", usando uma visão macroscópica e enxergando o projeto como um todo ou, numa visão microscópica, olhando para cada procedimento individualmente.

A nível macroscópico, os procedimentos são identificados por suas funções no projeto; assim é comum encontrar, por exemplo, procedimentos "explicativos", procedimentos "de controle", "super-procedimentos", procedimentos “básicos” (sub - procedimentos), etc. Assim, em geral, existe uma macro-estruturação boa: no nível mais geral (top) existe uma divisão dos procedimentos por funções.

3.2.1 Micro - estruturação

A nível microscópio observou-se que, em geral, há a divisão do problema em partes, gerando sub-procedimentos, quando a identificação de partes é imediata. Várias idéias foram utilizada:

- Idéia de "juntar pequenos pedaços da figura":

Observou-se uma pseudo-estruturação; isto é, o procedimento está "estruturado" apenas num primeiro nível. No nível mais "baixo", os procedimentos são "lineares", sem estruturação alguma em sua forma. Um exemplo:

```
ap avião
dt un mudepos [-40 30] mudedç 10
corpoav
asa
asalateral
cabine
fim

ap corpoav
pf 7 arcodir 2 180 pf 5 pe 90 pf 24 arcodir 2 180
pf 26 arcordir 2 90 pf 5
fim

ap asa
pe 90 pf 4 pt 6
fim

ap asalateral
un pe 180 pf 2 pd 90 pf 3 pe 90 pf 12 pd 90
pf 2 pe 90 ul pf 8
fim
```

```
ap cabine
un pe 90 pf 2 pe 90 pf 3 pd 90 ul
arcodir 2 180
fim
```

(R.R.)

Essa idéia de estruturar os procedimentos por função (conteúdo) levou a exageros que reforçam ainda o conceito de "pseudo-estruturação". Por exemplo, no procedimento "menino" (abaixo), pelo fato de "corpo" ser um membro de "menino", existe um procedimento separado só para definir "corpo", ainda que seja simplesmente PF 18.

```
ap menino
pernas corpo cabeça braços
fim

ap corpo
pf 18
fim

ap braços
pd 40 pt 15 pe 90 pf 3 pd 90 pf 15 pd 30
pf 15 pe 70 pf 5 pd 60
fim

ap pernas
....
```

(A.N.)

- Idéia de "desenhar" na tela e transcrever no procedimento básico:

Observou-se que os procedimentos básicos parecem feitos como uma transcrição do desenho feito na tela, no modo direto. Isso pode explicar o fato de não se encontrar estruturação alguma nos subprocedimentos. Escrever um procedimento estruturado envolve uma visão de conjunto e o modo direto representa a ferramenta mais primitiva para se "desenhar" com LOGO.

Reforça essa observação, também a ocorrência muito freqüente nos procedimentos, de seqüências do tipo un...ul para "ajeitar" a tartaruga em determinada posição na tela. O exemplo seguinte, mostra um "+" desenhado na tela como se faz no papel, sem levantar o lápis.

```
ap olho
pule 1
pf 4 pt 2 pe 90
pf 2 pt 4 pf 2
fim
```

(B.M.)

- Idéia de "usar procedimentos do texto como primitivas":

Ainda no nível microscópico de observação ocorreu com freqüência o uso isolado de procedimentos do texto adotado, num estilo muito diferente do estilo do aluno. Por exemplo, em projetos no tema 1, ("demonstrativo" de LOGO), são mostrados "murais" construídos de forma bem estruturada, usando como estrutura de controle o comando "repita", (como no texto), enquanto que os procedimentos de controle do projeto em si, usam estruturas de controle do tipo

teste...
severd...
sefalso... (vápara...)

Reforça essa observação, a falta de "generalização" observada em alguns projetos onde encontramos o procedimentos "poli", usado isoladamente como exemplo de recursão, ao lado de dois outros procedimentos para as figuras "circunferência" e "triângulo".

```
ap poli: passos :ângulo
se temcar então [se care = "f então [pare]]
pf :passos pd :ângulo
poli :passos :ângulo
fim

ap triângulo :lado
pd 30
repita 3 [pf :lado pd 120]]
pe 30
fim

ap circunferência : raio
tat
repita 36 [pf :raio * , 1745pd 10]
fim
```

(B.M.)

Esse exemplo mostra dois estilos diferentes de se "representar" polígonos. O recursivo, exemplificado por "poli", envolvendo como controle uma condição de parada. O iterativo, representado pelos procedimentos triângulo e circunferência, envolvendo como controle repetição

controlada. O estilo do aluno parece mais próximo da representação iterativa, onde não ocorrem generalizações. A idéia parece ser usar "ferramentas diferentes" (poli, triângulo), para gerar "objetos diferentes" (quadrado, triângulo), sem observar o fato de o processo de construção ser o mesmo.

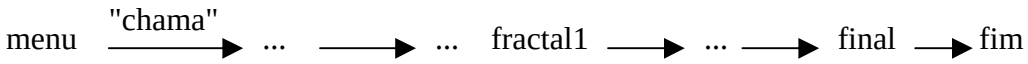
3.3 Controle

Pode-se observar, basicamente, três estilos diferentes de estabelecer o controle de execução nos projetos: linear, iterativo e recursivo.

3.3.1 Controle Linear:

No estilo linear o controle do fluxo de execução se dá ao longo do projeto, através de chamadas encadeadas de procedimentos e, em geral, o comando condicional é suficiente.

Exemplo:




```
ap final
esc [gostaria de ver outro desenho]? (s/n)]
coloque pri line "resp
se :resp = "s então [eltudo carregue "menu menu]
se :resp = "n então [esc[obrigado !...tchau]]
fim
```

3.3.2 Controle iterativo:

O fluxo de execução é determinado por procedimentos "de controle" envolvendo *loops* com estruturas de controle dos seguintes tipos:

1. condicional com teste isolado combinado a desvio incondicional:
   teste...
   severd... vápara...
   sefalso...

Exemplo:

```
ap comando
esc [...]
esc [...]
col ...
...
esc [se você deseja parar então [tecle "n] senão [tecle qualquer
outra letra]
teste care = "n
severd [vápara = "fim]
sefalso [pratique :nível + 1]
fim:
fim
```

A boa prática da programação estruturada não encoraja o uso de mecanismos de controle desse tipo, que podem levar a situações caóticas por ilegibilidade, em procedimentos mais sofisticados.

Note que as últimas quatro linhas do procedimento acima podiam ter sido substituídas simplesmente por:

```
se não care = "n então [pratique :nível + 1]
fim
```

O fato de não existir o operador "diferente" em LOGO aliado ao fato de o aluno sentir-se "desconfortável" com o uso de expressões booleanas, podem não tê-lo levado à segunda versão.

2. desvios condicionais:

O uso de desvios condicionais em geral é utilizado em sua forma mais simples (sem o "senão"), como selecionadores, em seqüência do tipo:

se ...então [...]
se ...então [...]

Testes redundantes parecem não ser notados. Não há preocupação de "facilitar para a máquina". Como exemplo encontramos:

```
ap final
se "pto1 > :pto2 então [A]
se :pto1 < :pto2 então [B]
se :pto1 = :pto2 então [C]
...
fim
```

em lugar de:

```
se :pto1>:pto2 então [A] senão [se :pto1<:pto2 então [B] senão [C]
```

Parece não haver uma tendência ao não uso de condicionais encadeados. Eles aparecem apenas em situações muito específicas como, por exemplo, em lugar de expressões lógicas:

```
ap movimento.carro1 :n
...
se coory <25 então [se coor x>-65 e coor x<-50 então [aviso pare]]
...
fim
```

que seria equivalente a:

```
se (coory < 25) e (coor x > - 65 e coor x < - 50) então [...]
```

3. repetição controlada:

repita...[....]

Repetição controlada em geral não é utilizada em procedimentos de controle, apesar de muito freqüente em procedimentos "básicos".

3.3.3 Controle Recursivo:

Controle realizado por procedimentos recursivos em geral utilizam recursão de ponta e funcionam como uma repetição controlada por um condicional, no "formato" a seguir:

```
ap xxx
se (condição) então [pare]

...
...
xxx
fim
```

4 Conclusões

O usuário mais freqüente de LOGO, parece ser a criança, o adolescente ou o adulto em sua primeira experiência no computador. O trabalho com universitários de uma área afim ofereceu uma oportunidade ímpar de se avaliar aspectos de LOGO, que, no caso anterior nem chegavam a ser explorados. Por outro lado, o trabalho em projetos relativamente extensos (cerca de 400 linhas) possibilitou uma visão um pouco mais ampla para o que se entende por "estilo de programação".

A análise dos projetos do tema 1 - "demonstrativo" do que é LOGO com base na reflexão do aluno sobre sua própria experiência com LOGO - gerou a base dos resultados aqui representados, confirmados em projetos dos demais temas. Dois aspectos devem ser colocados para reflexão:

1. Estilo x Conhecimento frágil x Fragilidades de Logo

Em literatura e nas artes em geral "estilo" está sempre associado à forma de expressão de um "expert". Não parece ser muito diferente quando a linguagem é de "programação". Assim, em se tratando de principiantes na "arte de programar", o estilo pode servir para mostrar as fragilidades conceituais do aluno.

Por outro lado, ao longo do trabalho de análise dos projetos procurou-se identificar características de LOGO que encorajam ou, ao contrário, dificultam, determinada forma de escrever os procedimentos. Citaremos alguns aspectos, a seguir:

a) Condicionais encadeados, ao estilo "Pascal" ficam ilegíveis em LOGO porque devem ser escritos em seqüência na linha.

Não existe formatação adequada, para tais tipos de construção em LOGO (veja exemplo a seguir). Entretanto , uma boa estruturação deveria simplificar essa forma de escrever.

```
ap listagem
...
se éelemento (pri:aux) :menu então [atr "list (jf:aux :list)repita
2 [esc []]] carregue pri:aux repita 2 [esc []] faça :aux esc [voce
quer desenhar de novo (s/n)?]
atr "resp care se (pri :resp = "n então [pare] senão [listagem]]
senão[esc[você errou algo,quer tentar de novo?(s/n)]atr "resp care
se (pri :res) = "n então [att listagem]]
fim
```

b) uso frágil de expressões booleanas combinado ao uso de condicionais leva a construções do tipo:

```
se éelemento :desenho :lista = "verd então [...]
```

(W.M.)

onde aparece o uso explicito de teste de igualdade para valor booleano.

```
teste évazia :comando severd [usuário1]
```
em lugar de `se évazia :comando então [usuário1]`

```
se coory > 34 [explosão queda]
se coory > 74 [explosão queda]
```
em lugar de `se (coory > 34) ou (coorx > 74) então [...]`

Nos dois primeiros casos os predicados parecem ser o elemento complicante.

Apesar das deficiências de programação, segundo o que a maioria concorda ser um "bom programa" note-se que a maioria dos alunos conseguiu realizar o projeto, independentemente da complexidade exigida por cada tema, dentro de seu próprio limite.

2. Estilo x Processo de aprender a Programar

Em geral notou-se grande preocupação com o efeito estético do resultado. Completa inobservância da estática dos procedimentos em si. Os projetos, com raras exceções parecem ter sido feitos exclusivamente para serem executados.

A observação do estilo de programar de iniciantes, mostra-nos um processo de "alfabetização" nesse novo meio de expressão e parece constituído por estágios análogos ao da própria evolução pela qual tem passado a metodologia de construção de programas.

Referências

Bentley, J. (1986). Programming Pearls. *Comm. ACM*. Vol. 29.

Knuth, D. E. (1984). Literate Programming. *The Computer Journal*. Vol. 27. nº 2.

LOGO. (1985). Manual de Referência. ITAUTEC. S.P.

Valente, J.A. Valente, A. (1985). *Logo - Conceitos, Aplicações e Projetos* (versão preliminar) editada posteriormente por McGraw Hill (1988).

Holanda, A. B. (1985). Novo Dicionário Aurélio.