Costa, C.E. (2006). Softwares para pesquisa: Relato de experiência. In O. Z. Prado, I. Fortim & L. Cosentino (Eds.), Psicologia & informática: Produções do iii psicoinfo e ii jornada do nppi (pp. 168-177). São Paulo: Conselho Regional de Psicologia de São Paulo.

SOFTWARES PARA PESQUISA: RELATO DE EXPERIÊNCIA

Carlos Eduardo Costa
Universidade Estadual de Londrina — UEL

DESENVOLVIMENTO DE *SOFTWARES* PARA PESQUISA: RELATO DE UMA EXPERIÊNCIA, SUAS DIFICULDADES E CUIDADOS NA PROGRAMAÇÃO

O uso de microcomputadores nos cursos de graduação, inclusive nos cursos de Psicologia, tem aumentado muito nas últimas décadas. Além do emprego de editores de textos e planilhas eletrônicas, muitos cursos utilizam microcomputadores para acesso a Internet, vídeo-conferências, análises estatísticas, aplicação de testes psicológicos, experimentação, etc. (cf. Anderson & Hornby, 1996).

Algumas pesquisas têm se preocupado com a determinação do uso ou não de microcomputadores em cursos de Psicologia e seus resultados subseqüentes. Anderson e Hornby (1996), por exemplo, investigaram as atitudes[1] positivas e negativas de estudantes de cinco cursos de Psicologia ante o computador. Os autores investigaram o quanto as atitudes dos estudantes poderiam ser influenciadas pelo gênero, idade, ano escolar, lócus de controle e experiência prévia com computadores. Os resultados demonstraram que a variável crucial é a experiência prévia dos alunos com computadores. Mais importante, as atitudes dos alunos em relação aos computadores mudaram ao longo do curso, de uma atitude negativa para uma positiva, quando eles utilizam os computadores em seu curso e a utilização de computadores em experimentação parece ter um efeito maior do que quando os computadores são utilizados, por exemplo, em aplicações de testes e análises estatísticas. Dada a facilidade resultante da utilização do computador, seja como instrumento de trabalho (como a confecção de relatórios, pesquisas na Internet, etc.), seja como instrumento de obtenção de dados de pesquisa, especialmente com humanos, o desenvolvimento e utilização de computadores e *softwares* em pesquisa, parece ter um efeito positivo sobre a formação do profissional.

Um fator que contribuiu para o aumento do uso de microcomputadores nos cursos de Psicologia é o menor custo de *hardware* em relação há três décadas[2]. Entretanto, se os *hardwares* estão mais baratos, o mesmo não se pode dizer do desenvolvimento de *softwares*[3]. O mercado para a venda de *softwares* especializados para pesquisa em Psicologia é muito restrito e os interesses particulares de determinados laboratórios tornam o mercado ainda mais inviável (Schneider, 1991). Considere-se, por exemplo, o interesse em pesquisas acerca do efeito de esquemas de reforço com humanos. O preço final de um *software* com recursos consideráveis para esse tipo de pesquisa pode não ser muito barato e encontraria pouco mercado (i.e., pesquisadores interessados na aquisição do *software* em número suficiente para viabilizar o investimento de profissionais especializados em sua programação).

Por isso, Beagley (2001) sugeriu que, cada vez mais, a formação em Psicologia deveria investir no desenvolvimento da habilidade de programação. Pesquisadores, especialmente, precisariam aprender a desenvolver *softwares* que supram suas necessidades de pesquisa. Essa, no entanto, é uma posição controversa. A habilidade em programação poderia ser mais útil para algumas áreas da Psicologia do que para outras. Se, nos cursos de graduação, fossem oferecidas tantas disciplinas quanto as demandas específicas que cada área exige, os currículos poderiam ficar inviáveis. Todavia, parece interessante que cursos de pós-graduação em *strito sensu* pudessem oferecer disciplinas optativas em linguagens de programação para pesquisadores ou incentivar que futuros pesquisadores invistam em cursos de programação.

[1] Os textos originais referem-se ao termo atitude, mas aqui o termo será entendido como "comportamento de aproximação e utilização" do computador enquanto ferramenta de trabalho.

[2] Essa afirmação aplica-se igualmente para todos os cursos universitários e todas as áreas, e não se aplica especificamente aos cursos de Psicologia. A ênfase que o presente trabalho atribui a Psicologia deve-se, na verdade, à audiência a que este trabalho se destina de maneira mais específica.

[3] Mesmo com boas iniciativas de *softwares* abertos e a possibilidade de se encontrar *softwares* de pesquisa em Psicologia gratuitos ou a custo baixos, no geral eles podem ser limitados em termos de recursos com relação às necessidades específicas de um pesquisador.

Apesar de o desenvolvimento de *softwares* úteis para pesquisa em Psicologia ter crescido nos últimos anos - o que pode ser informalmente verificado a partir de uma consulta das publicações de periódicos especializados como o *Behavior Research Methods, Instruments & Computers* -, a área parece ainda carecer de psicólogos com alguma habilidade em programação, especialmente no desenvolvimento de *softwares* para pesquisa (Beagley, 2001).

O desenvolvimento do *software* para pesquisas em Psicologia pode ter vantagens sobre a possibilidade de buscar financiamento externo para que uma empresa especializada desenvolva o projeto: a) seu custo financeiro é menor (embora o tempo despedido possa ser um fator negativo relevante); b) modificações que atendam as exigências de delineamentos específicos podem ser implementadas com menor custo financeiro e maior rapidez; c) problemas inesperados (*bugs*) que, eventualmente, possam ocorrer são corrigidos mais rapidamente. Outra vantagem é que um especialista em programação poderia ter soluções mais rápidas e eficazes para o desenvolvimento de partes específicas do *software*, entretanto, teria pouco conhecimento sobre o grau de controle e necessidades específicas no registro dos dados necessários em muitos pontos do projeto. Suprir o profissional em informática com todas essas informações consumiria muitas horas e privaria o pesquisador das vantagens apontadas anteriormente. Como afirmou Beagley (2001), a maioria dos psicólogos que desenvolvem *softwares* são programadores amadores, no entanto, um programador profissional seria - no melhor dos casos - um psicólogo amador.

Porém, lançar-se na programação de *softwares* para pesquisa não é uma tarefa fácil e exige a tomada de muitas decisões. Compartilhar essas dificuldades, sugerir caminhos e discutir alternativas pode ser um primeiro passo na direção da otimização de recursos no desenvolvimento de *softwares* de pesquisa. O presente trabalho parte das seguintes considerações: (1) o uso de microcomputadores tem aumentado muito nas últimas décadas nos cursos de graduação em geral e nos de Psicologia em particular; (2) o custo financeiro de *softwares* para pesquisa nem sempre é viável para muitos pesquisadores; (3) a contratação de profissionais especializados para o desenvolvimento de *software* pode trazer problemas quanto a prazos de entrega, correção de *bugs* e implementação de recursos adicionais. Isto posto, o presente trabalho tem por objetivo relatar a experiência e as dificuldades encontradas na programação de um *software* para pesquisa - a partir do relato da programação de um *software* para coleta de dados sobre programas de reforço com humanos, o ProgRef v3 (Costa e Banaco 2002; 2003)[4] — , discutindo os erros cometidos e caminhos mais promissores para aqueles que pretendam se aventurar pela difícil, mas instigante, área de programação. Nesse sentido, o trabalho volta-se principalmente para profissionais da área de Psicologia que pretendam desenvolver *softwares* para suas pesquisas, sem interesse eminentemente comercial no produto.

RELATO DE UMA EXPERIÊNCIA, SUAS DIFICULDADES E CUIDADOS NA PROGRAMAÇÃO

A escolha da linguagem de programação

Uma das primeiras decisões a serem tomadas diz respeito à escolha de uma linguagem de programação (Visual Basic, Delphi, Java, ASP, PHP, etc.). Alguns fatores deveriam ser considerados, não necessariamente na ordem em que são apresentados: (a) facilidade de aquisição do *software* de programação; (b) recursos de *hardware* que *software* de programação exige; (c) objetivo do *software* de pesquisa a ser programado (i.e., o *software* de pesquisa irá rodar em um computador específico ou trata-se de programação para Internet?); (d) história do programador com outras linguagens de programação: por exemplo, minha escolha pelo Visual Basic (VB) foi controlada, em parte, pela minha história de ter conhecimentos rudimentares da programação em BASIC; (e) Facilidade de encontrar cursos e programadores que possam lhe tirar dúvidas sobre a linguagem de programação que você escolher. Minha escolha não se mostrou muito feliz nesse aspecto. Na cidade em que resido encontra-se com alguma facilidade programadores e cursos de Delphi, mas não de VB; (f) Aplicação da linguagem de programação em outros contextos: minha escolha pelo VB

[4] O *software* ProgRef v3 foi programado em Visual Basic® 6.0 e é executável em microcomputadores do tipo PC em ambiente Windows®. A interface com o usuário (i.e., o experimentador) não requer que ele tenha conhecimento de linguagem de programação, apenas com os parâmetros necessários para o delineamento de uma sessão experimental em programas de reforço.

também teve influência da possibilidade de escrever programas em VBA (*Visual Basic for Applications*) que é uma linguagem utilizada para escrever macros no Excel®, por exemplo. Essa decisão foi muito útil porque a análise dos dados gerados pelo ProgRef foi bastante facilitada pelas macros que escrevi no Excel®.

O planejamento inicial

Um ponto de extrema importância, que eu ingenuamente ignorei, é o planejamento inicial do *software*. O pesquisador, antes de escrever a primeira linha de código, deve tentar responder algumas perguntas fundamentais:

(a) Para quem você está programando: Apenas você e sua equipe utilizarão o *software* ou ele será distribuído para outros pesquisadores? A resposta a essa pergunta irá determinar, entre outras coisas, quão amigável o *software* deverá ser.

(b) Para que você está programando: O *software* de pesquisa rodará em um único computador ou rodará na Internet? A resposta a essa pergunta ajudará na escolha de uma linguagem de programação adequada e nas habilidades de programação que devem ser adquiridas. Saber se você pretende programar para uma situação ou outra poderá ajudar na escolha de cursos a serem feitos no intuito de buscar a capacitação necessária para o desenvolvimento do *software*.

(c) Quais recursos de *hardware* você terá a disposição para rodar o *software*: Quando comecei a programar eu tinha claro que meu *software* deveria rodar em um ambiente Windows® 95 ou 98 em computadores Pentium 100 MHZ com 16 MB de RAM. Ou seja, meu *software* deveria poder fazer uso de computadores "obsoletos" que, muitas vezes, ficam "jogados pelos cantos" das universidades sem nenhuma utilização prática.

(d) O que seu *software* deverá fazer: gaste um bom tempo nisso. Escreva, em forma de texto, tudo o que seu *software* deverá fazer. Por exemplo, você poderia escrever algo como: "O *software* deverá ter uma tela inicial que pede ao pesquisador para inserir o nome, data de nascimento e código do participante da pesquisa. Depois que esses dados forem digitados..." Reveja muitas vezes esse texto e reescreva-o quantas vezes for preciso para que você tenha

uma descrição clara do *software*. Isso o ajudará a tomar decisões na hora da programação propriamente dita. Eu me arrependo amargamente de não ter feito isso. É muito mais fácil você inserir um trecho de código em seu programa se você escreveu a parte anterior prevendo o que viria depois. Apesar de um *software* poder ser programado em componentes independentes, você precisará "conectar" esses componentes para funcionar em um todo integrado. Quanto mais bem planejado o *software* melhor. É como construir uma casa, você pode pensar em cada cômodo isoladamente, mas se você for erguendo cada cômodo sem uma idéia do todo a construção poderá ser pouco funcional. Alguém poderá, no final das contas, ter de passar pelo banheiro sempre que quiser chegar à cozinha.

Estudar a maioria dos recursos do *software* de programação

Antes de começar a programar ProgRef eu estudei por algum tempo o VB. Eu estudava cada lição e fazia os exercícios sem pensar no *software* que escreveria. Eu pensava: "se eu começar a pensar no *software* que quero desenvolver, vou parar de estudar os recursos que poderão me ser úteis mais tarde". Eu estava certo. Porém, agi do modo errado. Parei de estudar os recursos e iniciei a programação do ProgRef. O resultado foi que em diversos pontos do *software* eu escrevia os códigos de uma maneira muito mais complexa e difícil de manter (i.e., de depurar). Se eu tivesse estudado mais os recursos de programação, eu, teria aprendido que havia uma maneira mais econômica e elegante de escrever determinados trechos do *software*. Por outro lado, não é útil ficar anos estudando todos os recursos que, talvez, você possa vir a utilizar. Novas versões do *software* de programação, como novos recursos, sairão antes de você terminar de estudar todos os recursos da versão que você utilizará. Por exemplo, se você não pretende desenvolver um *software* para Internet, talvez possa ser sensato pular capítulos de um livro que explica como usar a linguagem de programação para a Internet. Em outros momentos você precisará parar a programação do seu *software* para estudar um ou alguns recursos específicos para a solução daquela parte do projeto. Em outras palavras, é impossível você estudar tudo que você usará, mas é útil você estudar muito bem a linguagem de programação que você irá utilizar antes de começar.

Procure soluções "prontas" na Internet

Quando você se deparar com alguns problemas na implementação do seu código, procure na Internet por ajuda. Há diversos sites que fornecem o código-fonte pronto, para você recortar e colar no seu programa (e.g., www.deitel.com; www.codelines.com, etc.). Alguns exigirão alguma adaptação no código, mas mesmo quando esse é o caso a solução é muito mais rápida do que ficar quebrando a cabeça para escrever todo código.

Comente muito seu código-fonte

Toda linguagem de programação possui um recurso que lhe permite escrever comentários no código-fonte que não serão lidos na hora de compilar o programa. Esse recurso é muito útil, especialmente para programadores amadores. É comum você escrever um longo trecho de código e, duas semanas mais tarde, não se lembrar qual a função daquele trecho no seu programa. Não seja econômico nos seus comentários e não confie na sua memória.

Funcionalidade, elegância e interface com o usuário

Uma decisão que você terá de tomar é: o código-fonte do seu *software* deverá ser apenas funcional ou ele deverá ser funcional e "elegante". Há muitas maneiras de escrever um código que execute a mesma operação (i.e., tenha a mesma funcionalidade). Algumas maneiras são mais "elegantes", "limpas", do que outras. Soluções elegantes geralmente requerem mais reflexão e raciocínio lógico (além de conhecimento dos recursos do *software* de programação) e por isso podem levar mais tempo para serem planejados e escritos. Entretanto, uma vez escritos, eles são mais fáceis de verificar e menos propensos a conter erros. Algumas soluções que tomam 30 linhas de código podem ser escritas em 10 linhas. Gaste tempo pensando não apenas na solução (i.e., funcionalidade), mas também no modo de implementar a solução (i.e., na lógica de programação e recursos que podem ser utilizados para escrever o código de maneira mais elegante).

A interface é independente da elegância do código-fonte. Você pode ter um *software* escrito com elegância, mas a interface com o usuário não ser nada amigável. O ProgRef é um *software* que não primou, infelizmente, pela elegância do código-fonte, mas tem uma interface amigável com usuário. É relativamente fácil navegar pelas telas e uma explicação rápida ou uma leitura do artigo que descreve o *software* (Costa & Banaco, 2002) é suficiente para que o pesquisador programe sessões experimentais e colete dados. No outro extremo, eu poderia escrever um *software* cujo código-fonte é elegante, mas a interface com o usuário é pouco amigável. Uma vez utilizei um *software* que controlava caixas experimentais de condicionamento operante com ratos. Você precisava escrever, em um editor de texto, as variáveis relevantes para a coleta de dados em uma ordem específica que o *software* leria depois. Se alguém não lhe dissesse a ordem específica você não saberia programar uma sessão experimental. E, mesmo aprendendo a programar uma sessão experimental específica, você precisaria de novas instruções para programar uma sessão diferente. A decisão do quanto amigável a interface deve ser dependerá dos usos que você quiser dar ao *software* mais tarde.

Programação Orientada a Objeto

Um tipo de programação que o VB 6.0 não fazia adequadamente, mas que veio com força no VB.NET e está presente em diversas linguagens, é a Programação Orientada a Objetos (POO). A orientação a objeto usa classes para encapsular variáveis de instância (dados) e métodos (comportamento) (Deitel, Deitel & Nieto, 2002/2004).

O exemplo mais comum para falar de orientação a objetos é pensar em um carro. Um carro possui vários "objetos" como rodas, bancos, volante, câmbio, etc. Em vez de "programar o carro" como um todo, poderíamos programar os objetos isoladamente e depois implementar o código para que os objetos interajam de modos específicos. Uma vantagem da POO é que você pode utilizar objetos já programados para um *software* para ser utilizado em outro. Se eu fosse construir outro carro eu poderia utilizar vários dos objetos já construídos para o carro anterior[5].

Imagine, por exemplo, que eu tenho um *software* para o estudo de programas de reforço. Nesse *software*, um dos objetos poderia ser um cronômetro. O cronômetro é programado como um objeto e

[5] Outros conceitos úteis a POO são "herança" e "polimorfismo", mas esses conceitos não serão abordados aqui.

testado. Uma vez que o objeto esteja funcionando adequadamente não é preciso mais se preocupar com o código-fonte do objeto "cronômetro". É preciso enviar ao objeto a informação do momento de início e do momento final da contagem de tempo. Todo cálculo do intervalo de tempo é feito dentro do objeto cronômetro. Se eu resolver escrever um *software* para pesquisar, por exemplo, o tempo de reação eu posso utilizar o objeto cronômetro do outro *software*. Eu não preciso reescrever o código e nem me preocupar com sua funcionalidade (que já foi testada). Portanto, a POO representa economia de tempo na produção de novos *softwares* e facilidade na manutenção de códigos.

Considerações Finais

Espero que o presente texto seja um incentivo a alunos e profissionais para a programação de *softwares* de pesquisa e útil no sentido de evitar erros que eu cometi quando me aventurei por essa área. Nesse sentido, procurei expor no texto algumas poucas informações que eu mesmo gostaria de ter lido antes de começar meu trabalho de programação.

REFERÊNCIAS

Anderson, M. D. & Hornby, P. A. (1996). Computer attitudes and the use of computers in psychology courses. *Behavior Research Methods, Instruments & Computers*, 28(2), 341-346.

Beagley, W. K. (2001). Why we need more psychology programmers/ El Knife, a data utility for transforming spreadsheets. *Behavior Research Methods, Instruments & Computers*, 33(2), 97-101.

Costa, C. E. & Banaco, R. A. (2002). ProgRef v3: sistema computadorizado para coleta de dados sobre programas de reforço com humanos — recursos básicos. *Revista Brasileira de Terapia Comportamental e Cognitiva*, 4(2), 173-192.

Costa, C. E. & Banaco, R. A. (2003). ProgRef v3: sistema computadorizado para coleta de dados sobre programas de reforço com humanos — recursos adicionais. *Revista Brasileira de Terapia Comportamental e Cognitiva*, 5(2), 219-229.

Deitel, H. M., Deitel, P. J. & Nieto, T. R. (2004/2002). *Visual Basic. NET: como programar (C. Y. O. Taniwaki; F. L. P. Lucchini, Trad.)*. São Paulo: Pearson Education do Brasil.

Schneider, W. (1991). Equipment is cheap, but the field must develop and support common software for psychological research. *Behavior Research Methods, Instruments & Computers*, 23(2), 114-116.