Desenvolvimento de um Interpretador de Comandos e Avaliador Gráfico para o Ensino de Estrutura de Dados (IGED)

Dorgival P. da Silva Netto[1], Thiago José S. Oliveira[1], Tiago Davi N. de Sousa[2], Gilberto F. de S. Filho[1], Andrei Formiga[1,2], Alisson V. Brito[1,2]

Departamento de Ciências Exatas – DCE [1]
Programa de Pós-Graduação em Informática – PPGI [2]
Universidade Federal da Paraiba - UFPB

tiagodvneves@yahoo.com.br, {dorgival.netto, thiago.oliveira, gilberto, andrei, alissonbrito}@dce.ufpb.br

Abstract. *The problem of evasion in Computer Science undergratuate courses is well known, and the courses related to programming are among the main culprits. This paper presents the architecture for a programming environment called IGED (Interpreter for Data Structures), which allows the students to develop algorithms to manipulate data structures and visualize the execution of such algorithms by graphical animations. This paper presents the proposed architecture for IGED, a sample high-level program, and a prototype GUI for the tool.*

Resumo. *O problema da evasão nos cursos superiores da área de Computação é conhecido, sendo as disciplinas de programação uma das maiores responsáveis pelo alto nível de reprovação. Para auxiliar no ensino e aprendizagem das Estruturas de Dados e seus algoritmos, esse trabalho apresenta a arquitetura para um Interpretador Gráfico de Comandos Específicos para Manipulação de Estrutura de Dados, o IGED, que permite a elaboração de algoritmos de Estrutura de Dados e a visualização da execução desses algoritmos através de animações.Este artigo apresenta a arquitetura geral do sistema, um exemplo de programa de alto nível, e um protótipo da interface gráfica com o usuário do IGED.*

1. Introdução

No processo de formação dos alunos de cursos da área de Computação e Informática, uma das habilidades esperadas dos egressos é a de construir programas para solucionar problemas variados. Para que isto seja possível no ensino de Estrutura de Dados, o professor deve auxiliar o aluno a sistematizar e organizar a estratégia de solução para os problemas na forma de um algoritmo, analisando as possíveis estruturas de representação do problema que tragam melhor eficiência na sua execução. A formulação de um algoritmo geralmente consiste em um texto contendo comandos que devem ser executados numa ordem prescrita. O texto é uma representação concreta e tem um caráter estático, enquanto que a sua execução com um conjunto de valores iniciais é um evento dinâmico [Guimarães 1994]. Fazer com que o aluno compreenda esses dois aspectos complementares geralmente é um processo complexo, pois é preciso "visualizar" as estruturas dinâmicas das execuções dos algoritmos a partir da estrutura estática do texto do algoritmo.

O estudo de Netto (2010), que teve como objetivo investigar e identificar as dificuldades encontradas por alunos de cursos de computação no aprendizado da disciplina

Estrutura de Dados, mostrou que 33% dos alunos tiveram dificuldade com a abstração e 17% dos alunos consideraram a complexidade do assunto como uma barreira ao aprendizado. Considerando as dificuldades com conceitos específicos, 24% dos entrevistados responderam que o assunto de maior dificuldade foram as estruturas de listas, 22% dos participantes responderam ter mais dificuldade com o conceito de recursividade e 15% afirmaram serem as estruturas de filas as mais difíceis. Também foi identificado que a maioria dos professores da disciplina não utiliza ferramentas educacionais para auxiliar as aulas, com a possível exceção de ferramentas gerais de programação.

Com o objetivo de facilitar o entendimento dos conceitos abstratos envolvidos, acredita-se que o uso de Objetos de Aprendizagem que permitam a exploração dos conteúdos de forma digital e interativa, com recursos midiáticos como simulações, gráficos, desenhos e animações,, possa fazer com que os alunos tenham uma melhor compreensão dos conteúdos e se sintam mais motivados ao estudo.

Segundo Monteiro (2006), Objetos de Aprendizagem podem se tornar um valioso recurso pedagógico, com possibilidades de facilitar e tornar mais eficaz o processo de ensino. Considerando isto, e conhecendo-se a dificuldade de assimilação de conteúdo dos alunos nas disciplinas de Estrutura de Dados [Netto 2010], faz-se necessário o uso de novos recursos para apoiar o ensino e a aprendizagem desses conteúdos considerados de fundamental importância para a área de Computação e Informática.

Neste sentido, o presente trabalho propõe a arquitetura para um Interpretador Gráfico de Comandos Específicos para Manipulação de Estrutura de Dados, o IGED, permitindo ao aluno uma efetiva ferramenta de programação e acompanhamento de seus programas através de animações das abstrações de cada estrutura de dados abordada. Após uma discussão sobre as motivações do trabalho e da ferramenta proposta (Seção 2) e sobre os trabalhos relacionados na literatura (Seção 3), é apresentada a arquitetura geral para o IGED (Seção 4), e mostra-se um protótipo da interface proposta para o sistema (Seção 5). Finalmente, a Seção 6 conclui o artigo com as considerações finais. .

2. Uso de Animações de Algoritmos no Ensino

Segundo Garcia (2007), os livros-texto e aulas convencionais que abordam as estruturas de dados utilizam-se de mecanismos que vão além da descrição em (pseudo) linguagem de programação, como ilustrações. Entretanto, o excesso dessas ilustrações, quando vistas estaticamente, contribui muito pouco para aumentar a compreensão dos algoritmos, dado que além das informações que estão sendo descritas, há uma grande quantidade de comunicação implícita na correlação entre as imagens. Esta correlação pode ser mostrada através da animação gráfica de tais imagens, melhorando a comunicação da informação ao aluno ao deixar as relações entre as imagens explícita.

Em [Brown 1987], M. Brown introduziu a ideia de ambientes dedicados à construção de animações gráficas computadorizadas como ferramenta para ensino em computação, como já são aplicadas com sucesso em outras áreas, como química, anatomia e simulação de circuitos.

Um primeiro uso de animação de algoritmos é a criação de filmes [Hopgood 1974] ou sequências pré-programadas da execução de eventos [Brown 1987]. Essa abordagem é denominada passiva, pois o usuário se comporta como mero espectador e é útil para a complementação de livros-texto e aulas.

Uma limitação clara, no entanto, está no fato de o usuário não poder testar novos conjuntos de dados e ter que ficar limitado aos que são fornecidos pelos criadores da

animação. Para remover essa restrição, alguns sistemas [Brown 1991], [Stasko 1990], [Amorim 1993] introduziram facilidades para que seus usuários pudessem fazer as suas próprias experiências, direcionando o andamento do algoritmo. Tal abordagem é denominada ativa.

Resultados positivos para o uso dessa abordagem foram relatados em uma experiência realizada com o ambiente POLKA [Lawrence 1994]. O exame desses resultados confirma que a abordagem ativa é mais poderosa, pois tem-se uma participação maior por parte do usuário.

Uma nova abordagem para o uso de animações no processo de ensino e aprendizagem foi introduzida pelo ambiente Astral [Garcia 1997]. Nesse ambiente, o usuário pode visualizar animações de algoritmos que ele mesmo implementa, além de visualizar animações preparadas previamente. Essa nova abordagem, denominada pelos autores de abordagem construtiva, foi implementada e exemplificada no ambiente Astral, que possui uma interface de programação de animações em uma plataforma adequada e amigável.

3. Trabalhos relacionados

O sistema Balsa (Brown Algorithm Simulator and Animator) [Brown 1987] foi desenvolvido na Brown University no início dos anos 80 e seu intuito era o de servir como um laboratório de experimentação com representações dinâmicas de algoritmos em tempo real. Seu sucessor, Balsa-II, introduziu uma interface mais amigável e uma maneira homogênea para a execução e interação com animações, executando sobre a plataforma Macintosh e utilizando a linguagem Pascal para entrada dos algoritmos.

No início dos anos 90, surgiu o projeto Zeus de M. Brown [Brown 1991] cujas contribuições incluíam uma proposta de separação entre algoritmo e visualizadores, colocando interfaces bem definidas entre eles; o suporte à construção de programas de grande porte utilizando o paradigma da orientação a objetos e a utilização da linguagem Modula 3. Em Zeus, múltiplos algoritmos podem ser executados concorrentemente, facilitando a análise de diferenças e similaridades entre vários algoritmos.

Xtango [Stasko 1990] é um sistema de animação de algoritmos de propósito genérico que permite a construção de animações em tempo real. O objetivo principal é prover facilidades para os usuários que irão construir as suas animações, fornecendo uma interface de alto nível. Utiliza a linguagem C e opera sobre plataformas Unix e X11 Window System. Polka[1] é o sucessor de Xtango e foi projetado para dar suporte ao desenvolvimento de animações concorrentes, de maneira a facilitar a exibição de algoritmos paralelos.

O projeto Astral [Garcia 1997] tem como sua principal inovação a produção sistemática de animações, na qual o estudante utiliza uma coleção pré-estabelecida de procedimentos de manipulação gráfica das estruturas de dados para criar algoritmos mais complexos. O aprendiz pode construir a animação completa de uma dada tarefa através da composição desses procedimentos básicos, podendo assim facilitar a ligação do conceito abstrato com a programação do algoritmo.

A abordagem construtiva proposta no projeto Astral está baseada em procedimentos básicos de manipulação gráfica pré-programados, sendo os detalhes de sua implementação totalmente abstraídos do usuário através de uma biblioteca de programação. Entretanto, embora o Astral possibilite ao usuário a construção de suas próprias animações, sendo assim um avanço sobre as outras abordagens, o uso de componentes básicos pré-estabelecidos para

[1] http://www.cs.sunysb.edu/~algorith/implement/xtango/implement.shtml

criar os algoritmos e suas animações associadas também limita o que pode ser representado no sistema. Desta forma, a ferramenta IGED aqui proposta objetiva tornar a abordagem construtiva mais efetiva ao aluno, disponibilizando uma linguagem de programação completa que permita a animação de qualquer algoritmo. A arquitetura do IGED é descrita a seguir.

4. Arquitetura do Interpretador Gráfico de Estrutura de Dados.

A arquitetura proposta para o Interpretador Gráfico de Estruturas de Dados (IGED) é ilustrada na Figura 1. Nela podem-se visualizar três camadas distintas: Camada Gráfica, responsável pela iteração com o usuário; o Interpretador de Comandos que avaliará e executará os comandos passados pelo usuário; Camada Avaliadora, responsável por comparar o resultado obtido pelos comandos passados pelo usuário com o resultado esperado em cada exercício. Além das três camadas citadas, o sistema inclui um componente chamado de Gerador de Tarefas, responsável pelo cadastro das tarefas criadas pelos instrutores e disponibilização dessas tarefas para os usuários.

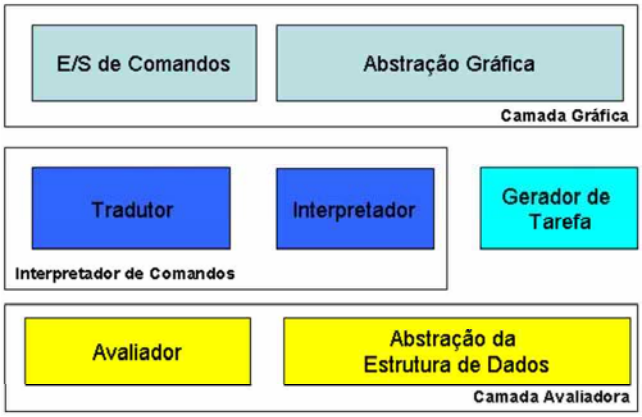

Figura 1. Arquitetura do interpretador Gráfico de Estruturas de Dados

4.1. Camada Gráfica

A Camada Gráfica, responsável pela interface com o usuário do sistema, é dividida em dois componentes: o primeiro é a E/S (Entrada/Saída) de Comandos, responsável por receber os comandos do usuário e entregá-los à camada Interpretador de Comandos (Entrada) e exibir os resultados dos comandos durante a execução de uma animação (Saída).

O segundo componente da Camada Gráfica, chamado Abstração Gráfica, é responsável por representar graficamente as estruturas de dados e suas operações, através do uso de animações. Esse componente deve se adaptar para as especificidades dos diferentes tipos de estruturas, podendo manipular estruturas estáticas como vetores ou dinâmicas como listas e árvores, e realizar tais manipulações através das ações básicas enviadas pelo componente Interpretador de Comandos.

4.2. Interpretador de Comandos

A camada responsável pela efetiva abordagem construtiva proposta neste trabalho é chamada Interpretador de Comandos, na qual se encontram dois componentes. O primeiro é o componente Tradutor, responsável pela tradução dos programas em linguagem de alto nível para uma linguagem intermediária; isso permite que o segundo componente, chamado Interpretador, execute o código intermediário para efetuar a manipulação das estruturas de dados. Os passos de computação realizados no nível da linguagem intermediária podem ser animados individualmente, e essa maior granularidade de passos básicos faz com que a animação se torne mais detalhada ao aprendiz, que tem assim uma experiência mais completa

dos seus passos e uma maior associação do código implementado ao seu efeito na abstração da estrutura de dados.

Para a implementação da camada Interpretador de Comandos, há a necessidade da definição de uma gramática para a linguagem de programação adotada pelo componente Tradutor. Também é necessário definir a linguagem intermediária que será usada na saída do Tradutor, pois essa linguagem define os passos básicos que serão executados pelo Interpretador e animados na Camada Gráfica. O fluxo das informações na ferramenta é ilustrado na Figura 2, iniciando com o programa de entrada do usuário, que é traduzido para os comandos da linguagem intermediária, que por sua vez são executados pelo Interpretador e animados pelo componente de Abstração Gráfica.

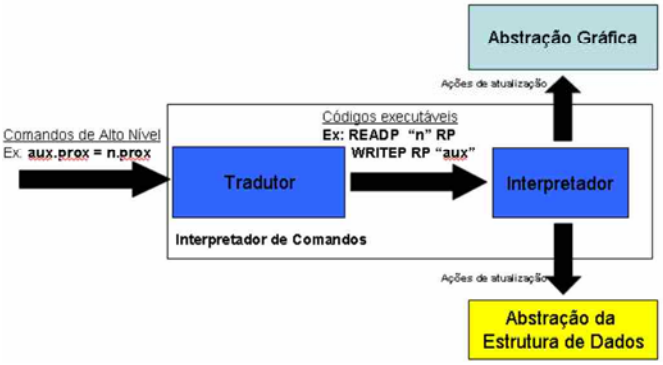

Figura 2. Fluxo de dados na camada Interpretador de Comandos.

Um dos benefícios diretos da divisão da camada Interpretador de Comandos em dois componentes é a flexibilidade que o sistema terá para trocar de linguagem de programação de alto nível, pela simples implementação de um novo componente Tradutor. Também é possível alterar a linguagem intermediária executada, embora isso tenha impacto em um maior número de componentes do sistema.

```
// Insere um número n em uma lista ordenada (crescente).
void InsereNo(int n, Lista<int> lista)
{
   var p = lista.frente;
   var q = null;

   while (p != null && p.valor < n) {
      q = p;
      p = p.proximo;
   }

   var novo = new Lista<int>.No;
   novo.valor = n;
   novo.proximo = p;
   if (q != null)
      q.proximo = novo;
   else
      lista.frente = novo;
}
```

Figura 3. Exemplo de programa na linguagem de alto nível do IGED.

Inicialmente, a sintaxe da linguagem de programação utilizada pelos usuários do sistema foi projetada para ser similar à linguagem Java e às linguagens da família C/C++, pois é um tipo de sintaxe com a qual a maioria dos alunos está familiarizado. Futuramente será possível implementar módulos Tradutores para outras linguagens de alto nível. Um exemplo de programa escrito na linguagem de alto nível do sistema IGED é mostrada na Figura 3. Este programa inclui apenas uma função chamada `InsereNo`, que insere um novo nó contendo um inteiro em uma lista encadeada que esteja ordenada em ordem crescente.

4.3. Camada Avaliadora

A Camada Avaliadora tem a responsabilidade de validar os comandos passados pelo aluno/usuário durante o processo de resolução de uma tarefa pré-cadastrada pelo instrutor no sistema. Logo, sua função é indicar ao aluno se a sua solução foi bem sucedida segundo a expectativa de resultado cadastrada anteriormente pelo instrutor.

Esta camada também está dividida em dois componentes: o primeiro componente é a Abstração de Estrutura de Dados, que contém a implementação de cada estrutura abordada e irá executar nestas estruturas os comandos recebidos e traduzidos pelo Interpretador de Comandos. Este componente irá trabalhar com duas instâncias idênticas da estrutura: a *original* e uma *réplica*. O objetivo da réplica é servir de controle da tarefa: o instrutor vai cadastrar um programa que serve de solução para a tarefa em questão; esse programa será executado sobre a réplica. Já a instância original receberá os efeitos da execução do programa inserido pelo usuário para resolver a tarefa.

O componente Avaliador terá acesso às duas instâncias (original e réplica) e irá comparar o seus estados finais. Se estiverem iguais, a resolução da tarefa será considerada com sucesso, caso contrário será considerada falha.

5. Protótipo de interface

Na Figura 4 é mostrado um protótipo da tela principal do IGED. Nessa tela, podemos identificar os campos: Exercício, responsável por mostrar a pergunta de um exercício previamente cadastrado, que deve ser resolvido pelo aluno; no centro da tela principal, encontramos o ambiente gráfico, responsável por apresentar a animação do código construído pelo aluno; na parte inferior da tela está a área Comandos, que é o campo onde o aluno escreverá o código para responder a questão mostrada no campo Exercício; no lado direito da tela, encontramos uma legenda, que mostrará o significado de todos os símbolos utilizados na animação para facilitar a compreensão da animação.

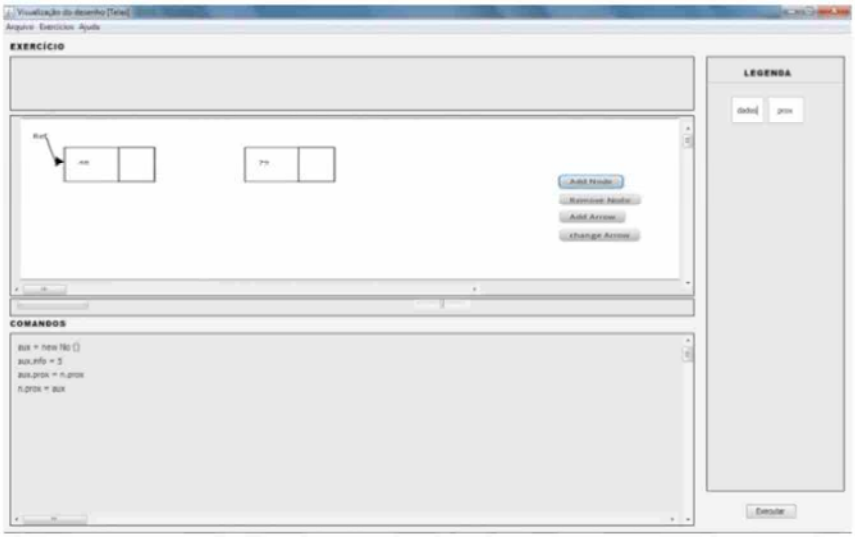

Figura 4. Protótipo da tela principal do Interpretador Gráfico de Estruturas de Dados

O JavaFX[2], foi escolhido para ser a plataforma utilizada para a construção da parte gráfica e as animações do IGED. Um fator importante para a escolha do JavaFX foi a possibilidade de desenvolver aplicações gráficas interativas de forma rápida e fácil.

Na Figura 5 é ilustrada uma animação feita em JavaFX que representa a inserção de nós em uma lista. O Passo 1 representa o estado inicial do ambiente, com dois nós, tendo o nó inicial (à esquerda) uma referência; no Passo 2, acontece a inserção de um novo nó, com um valor gerado aleatoriamente; o Passo 3 representa a ligação entre os nós através de setas; e por fim, o Passo 4 representa a ordenação gráfica dos nós, já que estes foram dispostos de forma desorganizada.

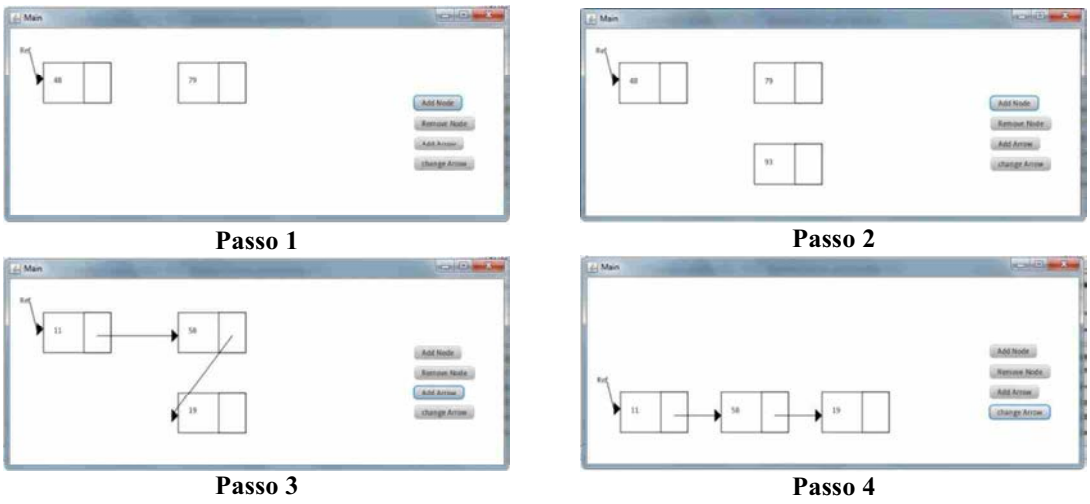

Figura 5. Representação de uma animação de Listas

6. Considerações finais e trabalhos futuros

O Interpretador Gráfico para Estruturas de Dados, IGED, é uma ferramenta proposta para uso no processo de ensino e aprendizagem de disciplinas dos cursos de computação que envolvem

[2] JavaFX: http://openjfx.java.sun.com/current-build/doc/reference/JavaFXReference.html

as Estruturas de Dados e seus algoritmos. Diferente de um livro-texto ou um conjunto de imagens estáticas, o IGED permite aos seus usuários uma abordagem ativa e construtiva, na qual os programas que manipulam as estruturas de dados podem ser construídos e seu resultado visualizado na tela imediatamente. Pretende-se que a visualização torne o assunto menos abstrato para os alunos, aumentando a retenção do conhecimento e diminuindo a evasão nas disciplinas relacionadas.

Os trabalhos futuros para o IGED compreendem a implementação de todas as camadas, inicialmente para as estruturas de dados mais simples como listas encadeadas. Tendo uma implementação terminada, será possível fazer um teste da ferramenta, através do uso da mesma em disciplinas relacionadas ao ensino das estruturas de dados. Em seguida, prevê-se a ampliação da implementação com novas estruturas e seguindo as sugestões que certamente surgirão do uso da ferramenta em sala de aula.

Referências

Amorim, R. V.; de Rezende, P. J. Compreensão de Algoritmos através de Ambientes Dedicados a Animação. XX Semish, 1993.

Brown M. H. Algorithm Animation. The MIT Press, 1987.

Brown, M. H. Zeus: A System for Algorithm Animation and Multi-View Editing. Proc. IEEE Workshop on Visual Languages, 1991.

Garcia, I. C.; De Rezende, P. J.; Calheiros, F. C. (1997). Astral: Um Ambiente para Ensino de Estruturas de Dados através de Animações de Algoritmos. In Revista Brasileira de Informática na Educação. Florianópolis: SC, Volume. 1, p. 71-80.

Guimarães, A. M.; Lages, N.A.C. (1994). Algoritmos e Estruturas de Dados. Rio de Janeiro , LTC.

Hopgood, F. Computer Animation Used as a Tool in Teaching Computer Science. Proc. 1974 IFIP Congress, pp. 889-892, 1974.

Lawrence, A. W.; Badre, A. N; Stasko, J. T. Empirically Evaluating the Use of Animations to Teach Algorithms. Technical Report GIT-GVU-94-07, Computer Science Department, 1994.

Monteiro, B. S., Cruz, H. P., Andrade, M., Gouveia, T., Tavares, R., Anjos, L. F. C.(2006) Metodologia de desenvolvimento de objetos de aprendizagem com foco na aprendizagem significativa, XVII Simpósio Brasileiro de Informática na Educação, Brasília.

Netto, Dorgival. Análise dos dados do questionário da disciplina Estrutura de Dados. Relatório Técnico. Departamento de Ciências Exatas, UFPB, 2010.

Stasko, J. T. Tango: A Framework and System for Algorithm Animation. Computer, 23(9): 14-36, sep 1990.