UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
BACHARELADO E SISTEMAS DE INFORMAÇÃO

ELIVANDO LIMA DE AMORIM

SELEÇÃO DE UM CONJUNTO DE ARTEFATOS EDUCACIONAIS PARA O ENSINO DE PROGRAMAÇÃO INTRODUTÓRIA

QUIXADÁ
2014

ELIVANDO LIMA DE AMORIM

SELEÇÃO DE UM CONJUNTO DE ARTEFATOS EDUCACIONAIS PARA O ENSINO DE PROGRAMAÇÃO INTRODUTÓRIA

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Graduação em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: computação

Orientadora Profª. Tânia Saraiva de Melo Pinheiro

QUIXADÁ
2014

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca do Campus de Quixadá

A543s Amorim, Elivando Lima de
Seleção de um conjunto de artefatos educacionais para o ensino de programação introdutória / Elivando Lima de Amorim. – 2014.
65 f. : il. color., enc. ; 30 cm.

Monografia (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2014.
Orientação: Prof. Dra. Tânia Saraiva de Melo Pinheiro
Área de concentração: Computação

1. Linguagem de programação (Computadores) 2. Algoritmos 3. Educação – Aprendizagem I. Título.

CDD 005.13

ELIVANDO LIMA DE AMORIM

SELEÇÃO DE UM CONJUNTO DE ARTEFATOS EDUCACIONAIS PARA O ENSINO DE PROGRAMAÇÃO INTRODUTÓRIA

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Graduação em Sistemas de Informação da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.

Área de concentração: computação

Aprovado em: _____ / dezembro / 2014.

BANCA EXAMINADORA

_____
Profa. Tânia Saraiva de Melo Pinheiro (Orientadora)
Universidade Federal do Ceará-UFC

_____
Prof. Dr. David Sena Oliveira
Universidade Federal do Ceará-UFC

_____
Prof. Dr. Rubens Fernandes Nunes
Universidade Federal do Ceará-UFC

RESUMO

Ensinar e aprender programação continuam sendo um desafio. O processamento mental de abstrações e o domínio de habilidades de resolução de problemas são as dificuldades mais comuns enfrentadas pelos estudantes. Assim, para trabalhar o raciocínio lógico dos alunos e para auxiliá-los no entendimento dos conceitos abstratos de linguagens de programação e algoritmos, esta pesquisa propõe um conjunto de artefatos educacionais, que são compostos por um jogo, exercícios de programação contextualizados, ferramentas de visualização e depuração de algoritmos e por uma linguagem de programação visual. A seleção desse conjunto se baseou nas principais propostas discutidas nos eventos nacionais de Informática na Educação, bem como nas experiências positivas relatas. Os resultados foram: a seleção de um conjunto de artefatos educacionais que pode ser utilizado, durante a maior parte de um curso de Programação Introdutória, como um recurso complementar às aulas; definição das dificuldades e necessidades dos alunos, tratadas por cada artefato educacional; e uma proposta de sequência de utilização do conjunto de artefatos educacionais selecionado.

Palavras chave: Programação. Algoritmos. Ensino. Raciocínio. Abstrações

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Um grande número de reprovações ocorre nas disciplinas de algoritmos e programação nos cursos relacionados à computação e à tecnologia da informação (VALASK; PARAISO, 2012; MARQUES et al., 2011; JESUS; RAABE, 2009). Isto acontece, em grande parte, pela dificuldade dos alunos iniciantes em lidar com o processamento mental de conceitos abstratos (AURELIANO; TEDESCO, 2012; MOTA et al., 2009) e, igualmente, em desenvolver o raciocínio lógico para a resolução de problemas (VALASK; PARAISO, 2012; PICCOLO et al., 2010; NETO, 2007). Devido a estes fatores, o ensino de programação torna-se um desafio, na medida em que implica na busca de novas alternativas para melhorar o nível de aprendizagem.

O ensino instrucionista, definido por Borges (2000) como o modo tradicional, não consegue motivar facilmente os alunos a se interessar pela disciplina de programação, pois no período inicial de formação não é claro para eles a importância de certos conteúdos (BORGES, 2000). Outras características negativas da abordagem tradicional são a exagerada repetição de problemas com enunciados textuais e a resolução dos mesmos em papel. Segundo Rapkiewicz et al. (2006), esta forma de explanação do conteúdo torna o processo de ensino e aprendizagem monótono e cansativo, além de não permitir aos estudantes verificar de forma eficiente a corretude dos algoritmos. Além da problemática dos métodos convencionais, o próprio perfil dos estudantes ingressantes requer atenção. Conforme Piva Jr e Freitas (2010), a maioria dos alunos que advém do ensino médio chega ao nível universitário sem uma base adequada nas disciplinas de português e matemática e apresenta dificuldades na interpretação de textos e na resolução de equações matemáticas.

Pelos problemas citados, percebe-se a necessidade do provimento de meios alternativos que auxiliem os estudantes no processo de entendimento dos conceitos abstratos e nas atividades de análise e resolução de problemas. Neste sentido, a literatura propõe o uso de ferramentas visuais para representar abstrações (SOUZA; GIRALD, 2011; HOSTINS; RAABE, 2007; BRITO JUNIOR, 2006; SANTOS; COSTA, 2005). Outra proposta interessante é o uso de jogos para abordar a questão da motivação e dos passos para a resolução de problemas (BATTISTELLA et al., 2012; COUTINHO et al., 2008; RAPKIEWICZ et al., 2006).

Ao analisar essas propostas, o estudo separou as dificuldades do aprendizado de programação em dois escopos de problemas: o primeiro engloba as dificuldades com raciocínio lógico e resolução de problemas, e o segundo envolve as dificuldades referentes à

compreensão das abstrações e à visualização mental das operações dos algoritmos. Guiada por essa classificação, e com a intenção de contribuir para a melhoria das experiências de ensino de programação, a pesquisa selecionou ferramentas e jogos, ao que denominou de artefatos educacionais. As ferramentas e os jogos, entretanto, não representam a proposta como um todo; há ainda outros componentes educacionais, dentre os quais, incluem-se exercícios contextualizados e linguagens de programação visuais. O termo **artefato educacional** será utilizado nesta pesquisa para representar genericamente quaisquer destes componentes educacionais que sejam considerados individualmente.

O objetivo deste trabalho é propor um conjunto de artefatos educacionais para auxiliar no ensino de uma disciplina de Programação Introdutória que consta na grade curricular de quatro cursos de graduação na área de Computação de uma Universidade Federal, visando auxiliar os alunos no entendimento das abstrações e no desenvolvimento das habilidades de interpretação e resolução de problemas. Para alcançar este objetivo, três etapas foram concluídas: na primeira foram identificados os tipos comuns de dificuldades no processo de aprendizagem de algoritmos e as principais necessidades dos alunos relatados na literatura; na segunda, fez-se um levantamento de artefatos educacionais que possam ser utilizados na disciplina programação introdutória; na terceira, do conjunto de artefatos educacionais levantados, selecionou-se um subconjunto de acordo com um determinado conjunto de requisitos de qualidade para ferramentas e jogos educacionais.

Com o conjunto de artefatos para auxílio do trabalho docente, pretende-se prover uma solução abrangente que combine e utilize os pontos fortes das melhores soluções de informática na educação encontradas para o ensino de algoritmos.

2 QUESTÕES FUNDAMENTAIS RELATIVAS AO ENSINO DE ALGORITMOS

Antes de se embarcar na busca por soluções para minimizar a complexidade das tarefas de ensinar e aprender programação, é necessário entender os conceitos fundamentais envolvidos e os principais entraves nessas atividades.

De antemão, entende-se que o conceito de programação está conectado à definição de algoritmo, e é uma atividade voltada para a solução de problemas, o que implica no profuso exercício de raciocínio lógico. Segundo Zanini e Raabe (2012), “dado o enunciado o aluno tem que construir um algoritmo, ou seja, descrever uma sequência de instruções que devem ser executadas pelo computador a fim de solucionar o problema. Assim, a forma como

os programas resolvem problemas exige dos programadores ao menos duas habilidades: a primeira é capacidade para interpretar problemas e decompô-los em partes menores, e a segunda é a capacidade para transformar soluções mentais em um conjunto de operações que podem ser executadas pelo computador. De acordo com Mendes et al. (2012), programar requer habilidades matemáticas, de representar bem os problemas e saber decompô-los, de criar modelos mentais que auxiliem a busca da solução.

O desenvolvimento dessas habilidades é o cerne da problemática de ensinar e aprender programação; por isso tem sido desenvolvidas técnicas, estratégias e jogos para auxiliar iniciantes de programação no desenvolvimento da habilidade de pensar e, paralelamente, tem-se construído também ferramentas de animação de algoritmos para melhorar a compreensão das operações executadas pelo computador.

Nesta seção, serão abordados os conceitos fundamentais necessários para o entendimento de como se dá o processo de ensino de algoritmos, levando-se em conta os desafios e problemas que existem nesse processo. Serão mostrados também o que tem sido proposto em eventos da área de Informática na Educação para facilitar a aprendizagem de programação, e quais as necessidades e dificuldades comuns dos alunos quando estão aprendendo a programar.

2.1 ALGORITMOS, ESTRUTURAS DE DADOS E PROGRAMAÇÃO

Um **algoritmo** é uma sequência finita de passos bem definidos e não ambíguos que objetiva a resolução de um problema e, por conseguinte, a realização de uma tarefa. Segundo Brito Junior (2006), um algoritmo, em linhas gerais, é qualquer conjunto finito de instruções que possa ser seguido para a realização de uma tarefa em específico ou para a resolução de determinado problema. Assim, receitas de bolos, instruções para montar um móvel e passos para achar uma saída são exemplos de algoritmos.

Quando processados pelo computador, algoritmos constituem programas em execução, os quais manipulam estruturas de dados específicas; sendo assim não é possível formular algoritmos sem que suas respectivas estruturas de dados sejam representadas. De acordo com Brito Junior (2006, p. 22), “para resolver um problema é necessário escolher uma abstração da realidade, em geral mediante a definição de um conjunto de dados que representa a situação real e a seguir, deve ser escolhida a forma de representar esses dados”. A escolha da representação dos dados é definida pela estratégia algorítmica que define um conjunto específico de operações sobre os dados.

Conforme apresentado, algoritmos existem, basicamente, para a resolução de problemas, os quais devem ser correta e integralmente compreendidos para que sejam adequadamente representados em algum modelo e solucionados. Assim, a construção de soluções surge da habilidade de interpretar questões e é justamente nesse quesito que se encontra grande parte das dificuldades dos alunos iniciantes. De acordo com Piva Jr e Freitas (2010), o baixo grau de leitura dos alunos advindos do ensino médio para o ensino superior tem como resultado menor capacidade para compreender problemas e lidar com abstrações.

Alguns autores matemáticos preconizam determinadas estratégias para a resolução de problemas em substituição ao modo aleatório de tentativa e erro. Segundo Dante (2002, p. 9), "problema é qualquer situação que exija o pensar do indivíduo para solucioná-la" e define ainda que um problema matemático "é qualquer situação que exija a maneira matemática de pensar e conhecimentos matemáticos para solucioná-la". Em outras palavras, o autor afirma que há formas ou estratégias adequadas para se chegar a soluções de maneira eficiente.

Polya (1975) relaciona quatro etapas principais para a solução de problemas:

- *Compreensão do problema:*

  Nesta etapa, o estudante considera o problema e define as partes principais do programa, as variáveis, os dados e o que de fato representa a solução;

- *Elaboração de um plano:*

  Fase em que se define, pelo menos de modo geral, quais os cálculos e operações sobre os dados (identificados na fase anterior) precisam ser executados para se obter o resultado. Neste momento, a ideia pode surgir gradualmente ou, então, após várias tentativas, aparecer repentinamente num lampejo mental.

- *Execução do plano:*

  Etapa em que se cria um roteiro para a execução do plano. Nesta ocasião o aluno deve ter atenção para que todos os detalhes e passos sejam inseridos no roteiro de forma que fique convencido da correção de cada passo do raciocínio.

- *Retrospecto:*

  Neste momento, faz-se o retrospecto do resultado final, percorrendo-se todo o caminho que levou a solução final. O objetivo é aperfeiçoar os passos que foram utilizados para a resolução do problema de maneira que as estratégias e o conhecimento utilizados sejam consolidados.

Pelo encadeamento dos conceitos estudados, define-se que programar é a atividade de construir e formular algoritmos usando técnicas e estruturas de dados para a resolução de problemas. Segundo Santos e Costa (2005), programação é uma atividade

voltada para a solução de problemas. Sendo assim, o código fonte torna-se elemento secundário; as prioridades são o entendimento do problema e o processo de concepção de sua solução. Reforçando esta conclusão, Santos e Costa (2005) afirmam que:

> o estudo de programação não se restringe ao estudo de linguagens de programação. As linguagens de programação constituem-se em uma ferramenta de concretização de produto de software, que representa o resultado da aplicação de uma série de conhecimentos que transformam a especificação da solução de um problema em um programa de computador que efetivamente resolve aquele problema. (SANTOS; COSTA; 2005, p. 3).

Ensinar programação, portanto, não é apenas ensinar uma linguagem de programação; sobretudo é enfatizar o uso de técnicas, formas e estratégias para a resolução de problemas. "Somente após o aprendizado dos conceitos de algoritmos e fundamentos de lógica, o estudante pode travar contato com uma linguagem de programação" (SANTOS; COSTA; 2005, p. 3).

2.2 DIFICULDADES NO PROCESSO DE ENSINO E APRENDIZAGEM DE ALGORITMOS

Pereira et al. (2005) relacionam algumas questões no processo de ensino e aprendizagem de programação:

1. Dificuldade para interpretar o problema antes mesmo da dificuldade para representá-lo em algum modelo formal.

2. Dificuldade para identificar os conhecimentos prévios para que os alunos tenham condições de aprender e desenvolver soluções algorítmicas de forma satisfatória. Segundo Pereira et al. (2005), alguns autores preconizam a necessidade de domínio de habilidades matemáticas prévias ou ao menos integradas, ou pelo menos desejável.

3. A necessidade de definir ou não o paradigma de programação para a estruturação das soluções dos problemas.

4. Dificuldade para se definir o nível e a maneira de formalização desejada, que pode variar entre o uso rudimentar ou de forma mais estruturada (pseudocódigo) de linguagem natural, uso de formas gráficas de visualização tais como fluxogramas e a utilização de linguagens de programação específicas.

5. Habilidades e experiências distintas dos alunos que determinam diferentes ritmos de aprendizagem. Esta situação também é salientada por Pimentel e França (2003):

   > Alunos não são iguais: possuem origens, experiências e habilidades diferentes. Isto explica em parte, o fato de alunos de uma mesma classe, submetidos às mesmas condições de ensino, apresentarem resultados distintos [Cardoso e Jandl 1998] e reforça a

> necessidade do uso de técnicas variadas que permitam ampliar os resultados de ensino. (PIMENTEL; FRANÇA, 2003, p. 2).

Tradicionalmente, o ensino de programação se baseia em aulas teóricas, apresentação de exemplos em pseudocódigo e posterior resolução de problemas (BRITO JUNIOR, 2006). Os exercícios de programação, geralmente, são apresentados fora de um contexto prático, o que dificulta a criação de imagens e modelos mentais que auxiliem na construção de algoritmos. Contextualizar significa, portanto, fornecer um ponto de partida para o aluno desenvolver o código dos algoritmos.

Outro problema que faz parte do escopo do processo de ensino tradicional é a forma de avaliação do aprendizado dos alunos. Segundo Pimentel e França (2003, p. 3), o fato de um aluno obter média cinco em uma disciplina, não indica que este aluno aprendeu metade do conteúdo; de acordo com os mesmos autores, "pode ser que ele tenha aprendido muito bem um tópico e quase nada de outros tópicos correlacionados". Pimentel e França (2003) salientam ainda o fato de as provas serem a principal fonte de diagnóstico do estado mental de cada aluno na disciplina e apontam para a necessidade de se desenvolver outras formas de avaliação do aprendizado.

Conforme Hu (2008), a geração que entra nas universidades é de alunos fascinados por jogos e interessados em desenvolver rapidamente seus próprios jogos. A experiência mostra que boa parte dos alunos iniciantes em cursos de Tecnologia da Informação, antes do contato com a disciplina de programação introdutória e, devido à convivência com sistemas interativos reais no seu dia a dia, tem a concepção de que desenvolverá sistemas similares logo no início de seu aprendizado de programação. Este cenário indica a necessidade de adequar o processo de ensino de programação às experiências cotidianas dos alunos, introduzindo ferramentas visuais que despertem maior interesse e fascínio. Coutinho et al. (2008) corrobora com esta ideia ao afirma que:

> A expectativa dos estudantes é que os programas devem se comportar da mesma forma que as interfaces web e os jogos digitais, com elementos gráficos bem elaborados e a alta interatividade, tornando o ensino de algoritmos baseado em programas de console é um tanto inadequado e desinteressante. (COUTINHO et al., 2008, p.3)

Outra questão é a própria complexidade em lidar com linguagens de programação. Segundo Pimentel et al. (2003, p. 2), no "aprendizado de programação, além de conceitos básicos existem conceitos difíceis de serem assimilados, como recursão e passagem de parâmetros". Uma vez que as operações definidas pelos algoritmos são executadas pelo computador, os programadores frequentemente pensam segundo a lógica da arquitetura de Von Neumann, ou seja, em termos de entrada, processamento e saída. Isto significa que

leitura e escrita em variáveis, chamadas de procedimentos e operações na memória do computador devem ser mentalizadas para a correta construção de soluções algorítmicas.

Esta habilidade de visualizar mentalmente as operações de máquina não é sólida na maioria dos iniciantes em computação; é uma competência que deve ser adquirida com esforço e determinação; porém, não deve ser fonte de demasiada fadiga a ponto de desmotivar o aluno.

2.3 PROPOSTAS MAIS COMUNS PARA O ENSINO DE ALGORITMOS

Segundo Santos e Costa (2006), diante da problemática do ensino de algoritmos, mudanças passaram a ser consideradas por graduados, mestres e doutores que trabalham com o ensino desta disciplina. Isto mostra a preocupação da comunidade acadêmica de computação em formar uma maior quantidade de profissionais de Tecnologia da Informação capaz de lidar com os constantes avanços tecnológicos e apta a prover as melhores soluções para os problemas apresentados pelo mundo real.

Aureliano e Tedesco (2012), por meio de uma revisão sistemática da literatura, analisaram os artigos sobre o processo de ensino-aprendizagem de programação para iniciantes publicados nos últimos 10 anos (2002-2012) no Simpósio Brasileiro de Informática na Educação (SBIE) e no Workshop de Informática na Escola (WIE), dois importantes eventos nacionais da área de informática educativa. Um dos resultados deste trabalho, apresentado no Quadro 1, corresponde à lista de alguns dos artefatos propostos para a melhoria do ensino de programação.

Quadro 1 – Lista parcial dos artefatos relacionados em Aureliano e Tedesco (2012)

Tipo de artefato	**Descrição**	**Total Artigos**
Avaliação de ferramenta	Avaliação de ferramenta de apoio à escrita cooperativa	1
Ferramentas de software	Ferramenta para suporte à programação em pares	19
	Ferramenta com simulação e visualização de código, correção automática	
	Ferramenta para acompanhamento e análise de programas	
Linguagem de programação	Linguagem de programação icônica	2
	Linguagem de programação para estudantes surdos	
Metodologia	Uso de jogos	7
	Estratégia para melhoria na abstração de resolução de problemas	
Técnica de avaliação	Estudo da taxonomia para elaboração de um instrumento de avaliação	1
		30

Fonte: adaptado de Aureliano e Tedesco (2012)

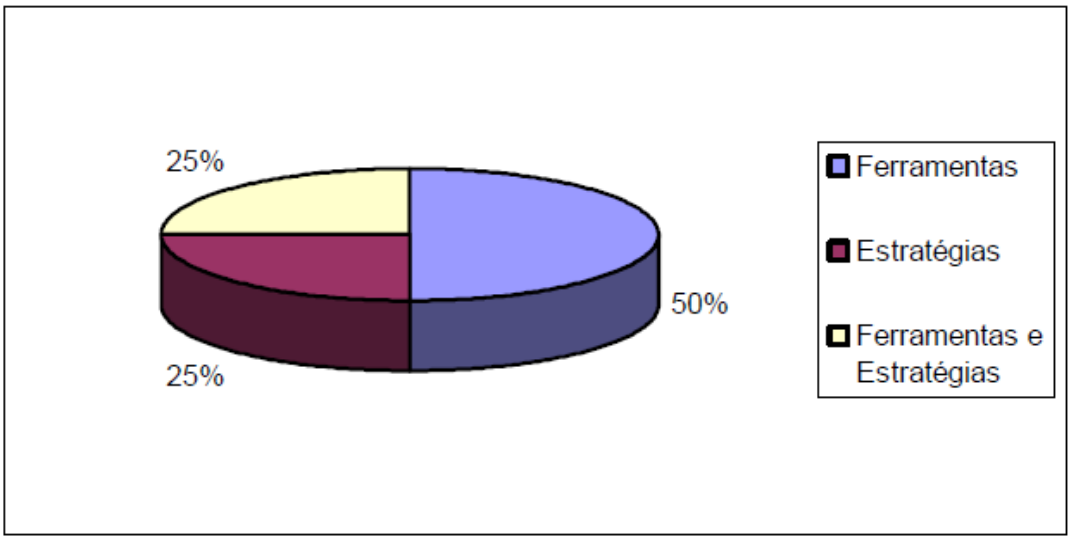
De acordo com as autoras, quase 65% dos 30 artigos selecionados propõem o uso de ferramentas de software para o processo de ensino-aprendizagem de programação para iniciantes. As metodologias correspondem a quase 23% dos artigos e a definição de novas linguagens de programação responde por quase 10% dos artigos. Outros tipos de artefatos representam os 6% restantes. Pela análise destes percentuais, percebe-se claramente a predominância de propostas de novas ferramentas de software, cujo uso deve se basear em alguma metodologia que explore o máximo dos recursos disponíveis.

Junior e Rapkiewicz (2004) também realizaram revisão sistemática do tema de ensino de programação em trabalhos publicados no Simpósio Brasileiro de Informática na Educação (SBIE) e no Workshop de Informática na Escola (WIE) entre os anos 1999 a 2003. Os autores classificaram as propostas se dividiram em três vertentes:

- *Ferramentas*: artigos que propõem a adoção de ferramentas computacionais para melhorar o processo de ensino.
- *Estratégias*: artigos que abordam estratégias de ensino e/ou formas de avaliação de competências.
- *Ferramentas e estratégias*: artigos que discutem estratégias baseadas no uso de ferramentas computacionais.

O Gráfico 1 indica a divisão percentual das vertentes:

Gráfico 1 – Percentual dos tipos de propostas para o ensino de programação



Fonte: Junior e Rapkiewicz (2004).

Brandão et al. (2012) descreve essas vertentes de produções científicas propostas por Junior e Rapkiewicz(2004) da seguinte maneira: as ferramentas são, principalmente, Sistemas Tutores Inteligentes para o aprendizado de programação ou para o gerenciamento de conteúdos das disciplinas; as estratégias incluem métodos de avaliação, como no trabalho de

Jesus e Raabe (2009), e métodos para a motivação; as ferramentas associadas à estratégias incluem trabalhos relacionados ao acompanhamento do aprendizado e à utilização de jogos como motivação.

Assim como a pesquisa de Aureliano e Tedesco (2012), o trabalho de Junior e Rapkiewicz (2004) revelou a preferência por ferramentas de software associadas à estratégias. Conforme estes autores (2004, p.5), "estratégias apoiadas por um ambiente virtual que proporcione a colaboração e ferramentas de desenvolvimento que permitam a execução e visualização de resultados parece ser uma melhor proposta".

Os resultados de Aureliano e Tedesco (2012) e de Junior e Rapkiewick (2004) corroboram com a proposta desta pesquisa de buscar tanto ferramentas de softwares, que permitam a visualização gráfica da execução dos algoritmos, quanto metodologias apoiadas no uso de jogos e exercícios contextualizados. Além disso, as pesquisas desses autores fornecem estatísticas que servem de base para uma mais acertada seleção do conjunto artefatos educacionais proposto por este trabalho.

2.4 OS DOIS ESCOPOS DE PROBLEMAS

É consenso entre os autores que é preciso buscar alternativas à abordagem de ensino de programação que se resume às aulas expositivas e à resolução de lista de exercícios. Nas publicações dos eventos nacionais de informática na educação, é comum a alusão às dificuldades dos alunos com abstrações (AURELIANO; TEDESCO, 2012; MOTA et al., 2009) e nas atividades de interpretação e resolução de problemas.

Levando-se em conta os relatos de problemas mais comuns, divide-se a problemática do processo de ensino e aprendizagem de programação em dois escopos: no primeiro aborda-se a dificuldade com o desenvolvimento do raciocínio lógico para a interpretação e resolução de problemas (PIVA JR; FREITAS, 2010; RAPKIEWICZ et al., 2006, PEREIRA et al., 2005); no segundo encontram-se as dificuldades com a compreensão de conceitos abstratos e para a visualização mental das operações dos algoritmos (AURELIANO; TEDESCO, 2012; MOTA et al., 2009; RAPKIEWICZ et al., 2006)

O **escopo de problema 1** reúne as questões relativas ao desenvolvimento do raciocínio lógico para a interpretação e resolução de problemas computacionais. As propostas em torno desse assunto visam aproveitar melhor a capacidade cognitiva dos alunos com técnicas que direcionem o pensamento para a solução de problemas. Piva jr (2010) cita, por exemplo, que é possível treinar a capacidade de abstração desenhando imagens que

representem a interpretação de um enunciado lido. Polya (1975) enfatiza o processo de solução de um problema, subdividindo-o em quatro etapas, de forma que o raciocínio tenha um maior campo de atuação. Outra forma de potencializar o pensamento cognitivo é aproveitar o que o aluno já conhece por meios de suas experiências cotidianas. Nesse sentido, a utilização de jogos tem sido indicada, pois além de ser parte da vida dos alunos, exercita as habilidades cognitivas e o desenvolvimento de estratégias (SALES; DANTAS, 2011, SAVI, 2011; RAPKIEWICZ et al., 2006 **)**. Outra proposta é a aplicação de exercícios de programação contextualizados, que representem tanto experiências de vida dos alunos como soluções para problemas do mundo real.

Levando em conta os problemas relativos ao escopo de problema 1 e uma abordagem de ensino prática e motivadora, a pesquisa considera o uso de **jogos** e de **exercícios contextualizado**s como recursos para auxiliar tanto o ensino dos fundamentais conceitos de programação como o desenvolvimentos de soluções. Esses artefatos educacionais trazem contexto prático significativo, permitindo aos alunos fazer associações mentais concretas que os aproximem das soluções dos problemas.

O **escopo de problema 2** aborda as dificuldades com o entendimento de conceitos abstratos e os problemas com a visualização mental das operações realizadas pelo computador. A forma como os algoritmos resolvem problemas, desdobrando-se em operações aritméticas e lógicas na memória do computador é muito diferente, por exemplo, da forma como as pessoas resolvem problemas matemáticos em papel. Por causa disso, o período inicial de aprendizado de programação exige do aluno um grande esforço mental para que ele se adapte ao novo paradigma de pensamento.

A despeito das dificuldades dos alunos em compreender abstrações, a evolução da computação caminha na direção de tornar as tecnologias mais acessíveis, de tal forma que os recursos tecnológicos sejam cada vez mais associáveis aos objetos e experiências da vida real. As linguagens de programação, por exemplo, evoluíram nesse sentido. De acordo com Kelleher e Pausch (2005), desde o início dos anos 1960, pesquisadores tentam desenvolver linguagens de programação e ambientes com a intenção de tornar a programação mais acessível para um maior número de pessoas. Esses autores afirmam que é necessário simplificar a sintaxe das linguagens, a fim de torná-las mais próximas da linguagem natural, facilitando assim a codificação.

No caso específico do ensino de programação introdutória, também se tem buscado maneiras de transformar as abstrações em elementos concretos. Para isso,

ferramentas de visualização de códigos (simuladores, depuradores) tem sido construídas para representar graficamente os dados e o comportamento dos algoritmos em execução (SOUZA; GIRALD, 2011; SANTOS; COSTA, 2005; HOSTINS; RAABE, 2007; PÍCCOLO et al., 2010). Assim o aluno deixa de ter apenas uma idéia abstrata dos algoritmos e passar a ter também uma representação visual concreta para auxiliar sua compreensão (SOUZA; GIRALD, 2011). Essas ferramentas visuais são feitas para permitir que os estudantes aprendam por experimentação, executando os programas e conferindo visualmente o resultado da execução de cada linha de código.

Ao considerar as recomendações de uso de ferramentas visuais para facilitar o entendimento de abstrações e do comportamento dos algoritmos, esta pesquisa admite **simuladores, debuggers e linguagens visuais** como artefatos educacionais indicados para combater os problemas relacionados ao **escopo de problema 2**.

2.5 CRITÉRIOS DE QUALIDADE DOS PRODUTOS DE SOFTWARE EDUCACIONAIS

Com exceção dos exercícios contextualizados, os artefatos educacionais definidos neste estudo constituem-se de produtos de software. Para selecionar as ferramentas e jogos é necessário entender o que é qualidade e como ela se traduz em interação e interface adequadas. Os critérios de qualidade de uso considerados neste trabalho correspondem a certas características elementares da interação e da interface que tornam o sistema adequado para produzir os resultados esperados por seus usuários.

2.5.1 Critérios de qualidade de sistemas de software

De acordo com Barbosa (2010, p.28), "a usabilidade é o critério de qualidade de uso mais conhecido e, por conseguinte, o mais frequentemente considerado". A norma sobre requisitos de ergonomia, ISO 9241-11 (1998), define usabilidade como sendo o grau em que um produto é usado por usuários específicos para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto de uso específico. O termo eficácia significa o lograr êxito em uma determinada tarefa e, eficiência, significa o quão prático, correto e inteligível foi o processo até o atingimento do objetivo. Usabilidade pode ser definida como um conjunto de fatores que permitem a uma pessoa interagir com um sistema de forma satisfatória. Barbosa (2010) considera os seguintes fatores de usabilidade:

- Facilidade de aprendizado
- Facilidade de recordação

- Eficiência
- Segurança no uso
- Satisfação do usuário

Como cada sistema possui características peculiares que o tornam único, o usuário precisa dedicar tempo para aprender a utilizar os diferentes softwares com que se depara. A avaliação da complexidade desse processo de aprendizado define o nível de **facilidade de aprendizado,** que, segundo Barbosa (2010), "se refere ao tempo e ao esforço necessários para que o usuário aprenda a utilizar o sistema com determinado nível de competência e desempenho". Uma vez tendo aprendido a utilizar um sistema, o usuário pode vir a esquecer a forma de utilizá-lo, o que ocorre, geralmente, por falta de contato contínuo com o mesmo. Nessa ocasião, a interface deve comunicar de forma clara as funcionalidades disponíveis e o modo de operá-las. A medida do esforço do usuário para lembrar como utilizar o sistema interativo a partir do que a interface lhe comunica determina o nível de **facilidade de recordação.**

A **eficiência** de um sistema interativo diz respeito à qualidade do processo que leva aos resultados esperados; em outras palavras, tem-se em vista se o atingimento dos objetivos é realizado de forma inteligível, pragmática, e em um tempo satisfatório.

No processo de aprendizado é natural que ocorram erros. Cada deslize cometido provoca receio, em maior ou menor escalar, dependendo das consequências; por causa disso, é recomendável que os sistemas tenham mecanismos de prevenção e recuperação de erros, de forma que os usuários se sintam tranquilos ao utilizá-los. É necessário, portanto, fornecer **segurança de uso,** que significa a proteção de um sistema contra condições desfavoráveis e até mesmo perigosas para os usuários (BARBOSA, 2010)

A tecnologia está sendo usada para apoiar quase todas as atividades humanas, não mais se focando em apoiar atividades de trabalho. Aplicações para entretenimento, educação, e saúde, por exemplo, são cada vez mais comuns e trazem à tona um conjunto muito maior de interesses. Além de buscar melhorias na eficiência e na facilidade de uso, os projetistas estão interessados nos efeitos do uso dos sistemas sobre os sentimentos e emoções das pessoas. Segundo Preece (2007, p.40), está existindo uma maior atenção para a criação de sistemas que sejam: satisfatórios, agradáveis, divertidos, interessantes, úteis, motivadores, esteticamente apreciáveis, incentivadores de criatividade, compensadores, emocionalmente adequados. Para Barbosa (2007), o fator de usabilidade relacionado com uma avaliação subjetiva que expressa o efeito do uso do sistema sobre as emoções das pessoas é chamado de **satisfação do usuário.**

O mesmo autor salienta ainda que esta definição é chamada por outros autores de **experiência do usuário**.

Os critérios de qualidade mencionados, com destaque para a usabilidade, fornecem para a pesquisa as diretrizes para a seleção de ferramentas de softwares com condições de uso satisfatórias, de tal forma que os estudantes não percam demasiado tempo com dúvidas sobre a operação dos sistemas. No caso de softwares educacionais não é diferente, exceto pelo fato de haver também fatores educacionais a serem analisados. Assim, as ferramentas de softwares educacionais pesquisadas foram analisadas heuristicamente, com base nos fatores de usabilidade, percorrendo-se a interface de cada uma delas.

2.5.2 Critérios de qualidade dos jogos educacionais

Segundo Savi (2011), existe uma dificuldade por parte dos professores em encontrar e utilizar bons jogos educacionais digitais e tais jogos ainda são pouco empregados. Salienta ainda que "isso ocorre, em boa parte, porque muitos jogos educacionais tem feito uso limitado de princípios pedagógicos e acabam sendo ignorados pelos educadores por agregarem pouco valor às aulas" (SAVI, 2011, p. 56).

O mesmo autor levanta a questão de que empresas especialistas no desenvolvimento de *games* criam produtos educacionais atraentes e divertidos, mas com pouco enfoque nos objetivos de aprendizagem. Por outro lado, comenta que jogos desenvolvidos por educadores com características mais acadêmicas e com pouco conhecimento da arte de criação de jogos, resultam na maioria dos casos em artefatos pouco atrativos. Deve haver, portanto, um equilíbrio entre pedagogia e diversão nos jogos educacionais. Em outras palavras, jogos educacionais devem cumprir um plano pedagógico bem definido e, ao mesmo tempo, proporcionar prazer e diversão.

Uma vez que requisitos de qualidade para jogos educacionais sejam levantados, é natural que se selecione algum modelo de avaliação para medir o quanto um jogo atende a tais pressupostos de qualidade. Porém, de acordo com Savi (2011, p. 99), a avaliação de jogos educacionais não é uma tarefa trivial e "há escassez de modelos especificamente voltados para a avaliação deste tipo de material instrucional".

É consenso entre os autores a afirmação de que os jogos devem possibilitar aprendizagem de forma divertida e desenvolver nos jogadores a habilidade para resolver problemas de maneira estruturada. Estas exigências dão uma ideia geral do modo como um jogo educacional deve ser avaliado. Faz-se necessário, portanto, quebrar estes requisitos em

características menores e mais específicas. Isto pode ser realizado, em parte, a partir da apresentação de Savi (2011) sobre os fatores, que, segundo ele, fazem com que jogos não atinjam as expectativas dos educadores:

- "Simplicidade de muitos jogos em relação aos *games* comerciais que não atendem as expectativas dos alunos já acostumados com a sofisticação dos jogos de entretenimento" (SAVI, 2011, p. 56).
  - Um jogo deve, portanto, fornecer um mínimo de riqueza gráfica e interatividade que não estejam tão distantes dos softwares comerciais.
- Tarefas muitos repetitivas, que tornam os jogos enfadonhos, ou muito pobres, que não permitem a compreensão progressiva dos conteúdos.
  – É desejável que o jogo possua uma diversidade de atividades de forma que exercite no jogador mais de uma habilidade.
- Relevância para currículo e precisão de conteúdo, ou seja, questões relativas à aderência aos conceitos que se quer ensinar.
  ***–** É recomendável que o jogo educacional forneça evidências claras dos conceitos que se propõe a ensinar e das habilidades que visa desenvolver.*
- "Saber como avaliar o progresso da aprendizagem dos alunos é outra questão que inibe o uso dos jogos pelos professores, especialmente no ensino online ou quando se classes com grande quantidade de alunos" (SAVI, 2011, p. 56).
  – É desejável que o jogo educacional possua mecanismos diretos ou indiretos que permitam medir o progresso de aprendizagem dos alunos.

Segundo Moratori (2003, p.24), "para uma utilização completa de um jogo educativo computadorizado é necessário realizar uma análise consistente sobre o mesmo, verificando tanto aspectos pedagógicos como aspectos de qualidade de software". O autor relaciona os seguintes requisitos pedagógicos em um jogo educacional:

- Objetivos bem definidos;
- Encadeamento lógico do conteúdo;
- Adequação do vocabulário;
- Possibilidade de formação de conceitos;
- *Feedback* apropriado;
- Possibilidade do professor incluir/ excluir/ alterar conteúdo do sistema.

Além dessas características pedagógicas, Moratori (2013, p.24) lista ainda os seguintes requisitos de qualidade técnica que devem ser observados:

- Execução rápida e sem erros;
- Interface amigável;
- Possibilidade de acesso à ajuda;
- Possibilidade de trabalho interativo;

- Possibilidade de controle do usuário sobre a sequência de execução do sistema;
- Possibilidade de correção de respostas;
- Possibilidade de sair do sistema a qualquer momento.

Na seção 2.5.1, foram apresentados critérios de qualidade para sistemas de softwares em geral. Esta seção, por sua vez, chamou a atenção para o fato de que a avaliação de jogos digitais educacionais não é trivial, já que o fator aprendizagem, que é difícil de mensurar, está envolvido. Dessa forma, ao analisar jogos, além de requisitos técnicos, a pesquisa considerou aspectos pedagógicos, adotando também uma abordagem heurística de percurso de interface, de modo a verificar o atendimento aos requisitos de qualidade.

3 PROCEDIMENTOS METODOLÓGICOS

A presente pesquisa tem natureza teórica, pois procura conhecer e aprofundar as discussões sobre o processo de ensino e aprendizagem de programação introdutória, com o intuito de descobrir as melhores propostas que tem sido adotadas para ajudar a melhorar o desempenho dos estudantes (VILAÇA, 2010). A partir dessa fundamentação teórica, o estudo pretende aplicar uma solução para um problema da realidade, caracterizando-se também como pesquisa aplicada. O processo metodológico empregado nessa pesquisa está dividido em três etapas:

Etapa 1: Discussão e apresentação dos artefatos educacionais

Na **seção 4**, são apresentados e categorizados os artefatos educacionais propostos pela pesquisa para tratar das questões relativas aos dois escopos de problemas criados a partir da revisão bibliográfica: dificuldade de desenvolvimento do raciocínio lógico para a interpretação e resolução de problemas; e dificuldades com a compreensão de conceitos abstratos para a visualização mental das operações dos algoritmos. Nessa etapa, com base nos estudos e experiências relatados nas publicações sobre o ensino de programação, são mostrados as razões e motivos pelos quais cada artefato educacional é vinculado a um dos escopos de problemas, de tal forma que fique claro seu potencial de contribuição.

Etapa 2: Vinculação dos artefatos educacionais às dificuldades e às necessidades no de processo ensino de programação introdutória

Nesta etapa, o estudo relaciona cada **categoria** de artefato educacional aos problemas e às necessidades frequentes dos iniciantes em programação. Depois disso, a pesquisa propõe uma sequência de utilização dos artefatos educacionais, com base na análise das publicações que apontam para uma introdução mais gradual à arte de programação,

envolvendo ferramentas visuais, uso de jogos e contextualização (MELO et al., 2011; MALAN; 2010).

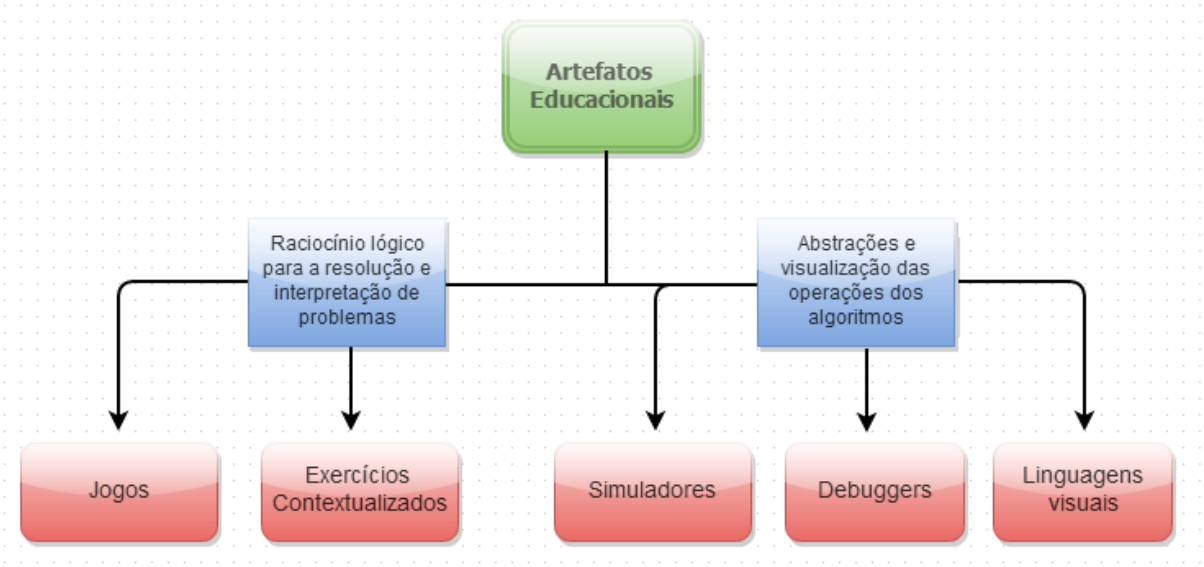
Etapa 3: Busca e seleção do conjunto de artefatos educacionais

Na **seção 6**, foram realizadas a busca e a seleção de cada artefato educacional, de acordo com os critérios de qualidade selecionados da **seção 2.5**, levando-se em conta também os experimentos de sucessos realizados por outros autores. Chegou-se, então, ao conjunto de artefatos final, construído para ser aplicado a maior parte do conteúdo da disciplina de Programação Introdutória estudada.

4 ARTEFATOS EDUCACIONAIS PARA OS DOIS ESCOPOS DE PROBLEMAS

Nesta seção, serão apresentados e discutidos as motivações para o uso de jogos e exercícios contextualizados para tratar das questões relativas ao escopo de problema 1, bem como as motivações para a utilização de simuladores, *debuggers* e de linguagens visuais didáticas para lidar com as questões relacionadas ao escopo de problema 2. Dessa forma, uma parte (categoria) dos artefatos educacionais é indicada ajudar no raciocínio lógico, e a outra parte é indicada para auxiliar no entendimento das abstrações e operações dos algoritmos. A Figura 1 resume como os artefatos foram categorizados:

Figura 1 – Categorização dos artefatos educacionais



Fonte: elaborado pelo autor

4.1 RACIOCÍNIO LÓGICO E INTERPRETAÇÃO DE PROBLEMAS

Nesta seção, serão apresentas as razões pelas quais jogos educacionais e exercícios contextualizados são indicados para estimular o desenvolvimento do raciocínio lógico e ajudar os estudantes a pensar estruturadamente. Conforme será discutido, o uso desses artefatos educacionais está baseado no aspecto motivacional e no fato de que eles trazem um contexto prático, transformando conceitos abstratos em concretos.

4.1.1 Jogos para o estímulo das habilidades de resolução de problemas

Segundo Savi (2011, p. 40), "um jogo é um tipo de atividade conduzida em um contexto de realidade imaginada, onde os participantes tentam alcançar ao menos uma meta, atuando de acordo com regras estabelecidas"; acrescentando-se nesta definição objetivos pedagógicos e a orientação de uma metodologia, tem-se um jogo educacional (SAVI, 2011). Conforme o mesmo autor, os principais benefícios do uso de jogos educacionais são:

- Os jogos possibilitam a aprendizagem fundamentada na experiência (*experience-based learning*), que maior interatividade, colaboração e aprendizagem em grupos;
- Potencialidade para uma aprendizagem mais efetiva através da aprendizagem ativa proporcionada pela interação com os objetos e personagens do jogo, em contraposição à abordagem passiva de absorção de informações transmitidas por um professor;
- Os jogos educacionais provocam o desenvolvimento de habilidades cognitivas, já que o jogador precisa elaborar estratégias para vencer o jogo e entender com os elementos do mesmo se relacionam. Dessa forma, também são desenvolvidas as habilidades de resolução de problemas, tomada de decisão, reconhecimento de padrões, criatividade e pensamento crítico;
- Jogos são eficazes para consolidar e aperfeiçoar as informações das aulas tradicionais por possibilitarem que os alunos apliquem na prática o que aprenderam;
- Jogos inserem a diversão, competição saudável e cooperação entre os alunos;
- Jogos tem o potencial para melhorar a capacidade dos alunos de trabalhar em equipe.

A utilização de jogos visa ao desenvolvimento do raciocínio lógico e da habilidade de formular estratégias para a resolução de problemas; além disso, busca promover a motivação e o engajamento. Conforme Tarouco (2004), os jogos podem ser ferramentas eficientes na medida em que divertem enquanto motivam, facilitam o aprendizado e aumentam a capacidade de retenção do que é ensinado, exercitando as funções mentais e intelectuais do jogador.

Conforme Rapkiewicz et al. (2006), os jogos promovem nos jogadores três atitudes fundamentais: decidir, escolher e priorizar. Segundo as autoras, todos os benefícios intelectuais do jogo resultam dessas ações fundamentais, que, em conjunto, ensinam a maneira certa de pensar e tomar decisões, pesando as evidências, analisando situações e consultando metas em longo prazo. Vemos aqui uma correlação com o método de Polya (1975) o qual implementa a resolução de problemas em quatro etapas: compreensão do problema, estabelecimento de um plano, execução do plano e retrospecto. Esta abordagem visa também ao aprendizado de uma forma eficiente de pensar e raciocinar.

Um dos principais benefícios dos jogos como ferramentas de construção do conhecimento é a motivação, tendo em vista que fazem parte da realidade prática vivenciada pelos alunos. Dessa forma, os alunos elevam sua atenção e tornam-se mais engajados durante o desenvolvimento das tarefas, e quanto maior a interatividade e riqueza gráfica dos jogos utilizados mais estes resultados se fazem notáveis. Segundo Rapkiewicz et al. (2006):

> Os jogos com atividades para exercitar a habilidade mental e a imaginação, agradam, entretêm, prendem a atenção, entusiasmam e ensinam com maior eficiência, porque transmitem as informações de várias formas, estimulando diversos sentidos ao mesmo tempo e sem se tornar cansativo. (RAPKIEWICZ et al., 2006, p. 5).

Outro fator importante é a simulação. O aluno acompanha a sua ação sobre o jogo e confere os resultados de suas decisões e, dessa forma, consegue modificar ou melhorar sua estratégia para alcançar o objetivo pretendido. “O jogo permite tornar real o que seria apenas imaginário no caso de enunciados de problemas apresentados no quadro ou em listas de exercícios no papel” (RAPKIEWICZ et al., 2006, p. 4).

Conforme citado, jogos são apontados como excelentes ferramentas para o exercício do raciocínio lógico e para estimular as habilidades de resolução de problemas. Há outras alternativas para isso, como, por exemplo, exercícios de interpretação de enunciados (PIVA JR; FREITAS, 2010), porém o uso de jogos é uma boa escolha porque são de grande interesse do público jovem, o que já é muito vantajoso, haja vista o engajamento que promove. Considerando essas vantagens, a pesquisa propõe a utilização de jogos para ajudar os alunos nas dificuldades relacionadas ao escopo de problema 1.

4.1.2 Exercícios contextualizados

“É comum que exercícios de programação resumam-se a ‘calcule isto’ e ‘resolva aquilo’, onde o resultado do cálculo ou da resolução não é utilizado para algo que faça sentido dentro de um contexto maior” (JESUS, 2010, p.2). Por causa disso, tanto o aluno é deixado para processar as abstrações sem auxílio extra, como fica desinformado acerca de como a

computação é aplicada na vida real. Por outro lado, tendo por base o fato de que a compreensão é uma das etapas do processo de construção dos algoritmos (ZANINE; RAABE, 2012), fornecer um contexto prático é uma forma de facilitar o entendimento, uma vez que isso cria uma correlação da abstração com a realidade. Esse entendimento é embasado no construtivismo, segundo o qual, "as pessoas constroem os seus conhecimentos, a partir de uma intenção deliberada de fazer articulações entre o que conhece e a nova informação que pretender absorver" (TAVARES, 2004).

De acordo com Pelizzari (2003, p.9), a aprendizagem é "significativa à medida que o novo conteúdo é incorporado às estruturas de conhecimento de um aluno e adquire significado para ele a partir da relação com seu conhecimento prévio". Do contrário, a aprendizagem se torna mecânica ou repetitiva, como o conteúdo sendo armazenado isoladamente ou por meio de associações arbitrárias na estrutura cognitiva (PELIZZARI, 2003).

Segundo Zanini e Raabe (2012), frente ao enunciado de um problema o aluno, independemente de qual estratégia utilize, passa por três etapas: (i) compreensão do problema (abstração dos dados); (ii) elaboração de uma sequência lógica de instruções (solução proposta); (iii) avaliação e depuração da solução proposta. De acordo com os autores, todas estas etapas dependem do correto e detalhado entendimento do enunciado, o que, possivelmente, pode ser facilitado pela contextualização.

Zanini e Raabe (2012) extraíram 428 enunciados dos livros adotados nas disciplinas introdutórias de programação nos cursos de ciência da computação no Brasil. Os enunciados analisados abrangem desvios condicionais e lações de repetição. O

Quadro 2 exemplifica os tipos comuns de enunciados encontrados pelos autores nos livros de programação.

Quadro 2 – Tipos comuns de enunciados

1 – Apresentar os quadrados dos números inteiros de 15 a 200.
2 – Apresentar o total da soma obtida dos cem primeiros números inteiros (1+2+3+4+...+98+99+100).
3 – Escreva um programa que apresente a série de Fibonacci até o décimo quinto termo. A série de Fibonacci é formada pela sequência: 1, 1, 2, 3, 5, 8, 13, 21, 34, ..., etc. Esta série se caracteriza pela soma de um termo atual com o seu anterior subsequente, para que seja formado o próximo valor da sequência. Portanto começando com os números 1, 1 o próximo termo é 1+1=2, o próximo é 1+2=3, o próximo é 2+3=5, o próximo 3+5=8, etc.

Fonte: Zanini e Raabe (2012)

O primeiro enunciado do é objetivo, pois apresenta o que deve ser feito de forma direta, e não possui qualquer exemplificação. O segundo é objetivo e apresenta uma ilustração para facilitar o entendimento. O terceiro, assim como os outros, é objetivo em um contexto matemático, e apresenta exemplificação e o processo de resolução.

Nos exemplos citados, nota-se a predominância de problemas de contexto matemático fora da realidade prática de muitos alunos, especialmente daqueles cuja base matemática não foi consolidada no ensino médio. A pesquisa de Zanini e Raabe (2012) verificou justamente essa questão: a predominância de problemas objetivos e descontextualizados. Para ser mais específico, dos 428 enunciados analisados pelos autores, 64,72% (277) apresentam o problema de forma objetiva e apenas 35,28% (151) de forma contextualizada. Dentre os contextualizados, mais da metade desses exercícios contextualizados é puramente matemática, o que pode ser visto mais detalhadamente no Quadro 3.

Quadro 3 – Categorização dos exercícios contextualizados mais comuns

Contexto	%	Descrição
Matemático	55,14	O foco do enunciado é a resolução de problemas puramente matemáticos, sem que este seja aplicado a outro contexto.
Comercial	6,31	O foco do enunciado é a manipulação de produtos, mercadorias, como compra e venda, cálculo de comissão, prestação de serviço.
Pessoa	5,61	O foco do enunciado é trabalhar com elementos relacionados diretamente a uma pessoa como peso, altura, sexo.
Escolar	4,91	O foco do enunciado é resolver algum problema escolar, como cálculo de média, verificação de aprovado ou reprovado.
RH	4,21	O foco do enunciado é resolver algum problema relacionado a recursos humanos, como cálculo de salário, dados relacionados a funcionários, cálculo de gratificação, recrutamento e seleção de

Fonte: Zanine e Raabe (2012)

Em contraposição a abordagem eminentemente matemática, a OBI (Olimpíada Brasileira de Informática) apresenta problemas que refletem situações comuns. A maioria das questões deste campeonato começa com um relato de uma situação cotidiana para apenas depois apresentar desafios de algoritmos a serem desenvolvidos. Essa abordagem tem o custo de tornar os enunciados bem maiores que os descontextualizados, o que deve ser visto de forma positiva, tendo em vista que o aluno exercita de forma mais intensa a leitura e passa a enxergar mais claramente a computação na vida real. O Quadro 4 apresenta um dos problemas de uma edição da OBI:

Quadro 4 – Problema JPNEU do SPOJ

Problema: JPNEU

Calibrar os pneus do carro deve ser uma tarefa cotidiana de todos os motoristas. Para isto, os postos de gasolina possuem uma bomba de ar. A maioria das bombas atuais são eletrônicas, permitindo que o motorista indique a pressão desejada num teclado. Ao ser ligada ao pneu, a bomba primeiro lê a pressão atual e calcula a diferença de pressão entre a desejada e a lida. Com esta diferença ela esvazia ou enche o pneu para chegar na pressão correta. Sua ajuda foi requisitada para desenvolver o programa da próxima bomba da SBC - Sistemas de BombasComputadorizadas

Tarefa

Escreva um programa que, dada a pressão desejada digitada pelo motorista e a pressão do pneu lida pela bomba, indica a diferença entre a pressão desejada e a pressão lida.

Entrada

A primeira linha da entrada contém um inteiro N que indica a pressão desejada pelo motorista ($1 \leq N \leq 40$). A segunda linha contém um inteiro M que indica a pressão lida pela bomba ($1 \leq M \leq 40$).

Saída

Seu programa deve imprimir uma única linha, contendo a diferença entre a pressão desejada e a pressão lida.

Exemplo Entrada	Saída
30	12
18	

Fonte: adaptado Site Spoj

O contexto desse exemplo, juntamente com a clareza de suas especificações de entradas e saídas, dá ao aluno um significado para o trabalho mental que irá realizar, fazendo-o entender que seu algoritmo de fato resolve um problema da vida real.

Levando em conta que exercícios contextualizados tem o potencial tanto para motivar como para facilitar o entendimento dos problemas, a pesquisa considera a aplicação desses artefatos educacionais como um recurso para auxiliar os alunos nas dificuldades relacionadas ao escopo de problema 1.

4.2 ABSTRAÇÕES E VISUALIZAÇÕES

Nesta seção, serão discutidas as razões pelas quais linguagens de programação visuais e ferramentas de simulação e visualização de códigos podem diminuir as dificuldades dos alunos com conceitos abstratos. Conforme será discutido, essas dificuldades tem a ver com a sintaxe das linguagens de programação e com o entendimento do processo de execução dos algoritmos.

4.2.1 Linguagens de programação visuais

As pessoas lidam frequentemente com algoritmos em seu cotidiano, resolvendo problemas de forma estruturada e sequencial. As atividades diárias de uma empresa moderna, por exemplo, são baseadas em processos bem definidos e ordenados, determinando assim

scripts ou programas de execução. Esses scripts de processos são sustentados por delegação de tarefas que são atribuídas a cada pessoa em sua função.

As tarefas diárias e os processos das empresas são, portanto, algoritmos com os quais as pessoas estão acostumadas, ao passo que algoritmos para programas computacionais representam uma mudança de paradigma de pensamento, pois é necessário o aprendizado de uma linguagem de programação que o computador entenda. De acordo com Valentim (2009):

> A linguagem de programação pode ser definida como um conjunto de regras e comandos com sintaxe e semânticas próprias, utilizadas pelo programador para traduzir os algoritmos, e tem como objetivo enviar comandos para serem executados pelo processador e memória do computador (VALENTIM, 2009, p. 4).

Aprender uma nova linguagem, seja computacional, seja um idioma, é sempre um desafio, ainda mais quando se trata de uma linguagem distante da usual. Para Valentim (2009) o aprendizado de uma linguagem de programação torna-se difícil, devido à sobrecarga de sintaxe que deve ser assimilada pelo aluno. Corroborando com essa visão, Mendes et al. (2012) afirma que a sintaxe é a principal dificuldade encontrada pelos programadores.

Com o intuito de inserir estudantes na arte de programação, facilitando a transição do pensamento usual para o computacional, a literatura tem relatado experiências positivas sobre o uso de linguagens visuais baseadas em blocos de construção para garantir que os alunos se concentrem muito mais na lógica de programação do que na sintaxe da linguagem, construindo assim algoritmos através de um sequência de comandos, num contexto similar à delegação de tarefas para a conclusão de uma atividade estruturada. (MELO et al, 2011; MENDES et al, 2012; SANTOS NETO, 2013; RIBEIRO, 2012). “Com uma interface visual e interativa, estas ferramentas buscam tornar a aprendizagem mais divertida e motivadora” (VALASK e PARAISO, 2012). Outro objetivo importante no uso de linguagens visuais, é diminuir a carga cognitiva com abstrações e sintaxe (RIBEIRO, 2012). Existem vários ambientes de desenvolvimento com base em programação visual como, por exemplo: Scratch, Alice, Greenfoot, Lego Mindstorms, StageCast, Ivprog.

4.2.2 Ferramentas de simulação e visualização de algoritmos

Para compreender a execução dos algoritmos é preciso assimilar as operações do computador que estão sendo realizadas, tais como leitura e escrita de variáveis, laços de repetição e tomadas de decisão. Soares et al. (2004) citam, por exemplo, que na implementação de algoritmos em grafos alunos normalmente criam, mentalmente ou através de desenhos, visualizações da execução desses algoritmos na tentativa de entender seus

passos; além disso, salienta que grande parte dos alunos sente dificuldade nesse tipo de implementação pela falta de suporte de ferramentas de visualização e interação com grafos.

Em aulas expositivas, o professor geralmente gesticula e movimenta as mãos para construir um modelo do conceito que está explicando. Este método para se fazer entender nem sempre alcança seu objetivo e muitos alunos acabam por não conseguir formar a imagem conceitual pretendida pelo professor. Neste tipo de abordagem, o esforço cognitivo é muito maior, se utiliza mais tempo e a aula se torna mais cansativa. Rapkiewicz et al. (2006) explicita esta situação ao afirmar que "para a compreensão das operações realizadas pela máquina, exige-se um grande esforço cognitivo do professor para explicar e para o aluno abstrair e entender como ocorrem internamente essas operações até chegar o resultado esperado"

Diante destes problemas, torna-se necessário a busca por mecanismos mais eficientes para a representação dos conceitos e abstrações. Uma alternativa viável é o uso de ferramentas de simulação (ou de animação) e depuração de algoritmos. Conforme Santos e Costa (2005), o intuito da adoção de produtos de software é procurar transformar processos abstratos em concretos, de forma que haja melhor compreensão dos tópicos abordados. Corroborando com esta visão, e segundo Souza e Girald (2011),

> Uma ferramenta computacional para fins educativos pode trazer inúmeras vantagens para alunos e professores. Um professor ganha um aliado para ensinar, o aluno tem uma ferramenta que apresenta os conceitos abstratos de uma forma mais didática, sem contar a economia de tempo que pode ser conseguida com o apoio de uma ferramenta educacional. (SOUZA; GIRALD, 2011, p. 3).

Ferramentas para auxiliar no entendimento de conceitos abstratos geralmente são construídas para permitir simulação e/ou depuração do algoritmo. (SOUZA; GIRALD, 2011; SANTOS; COSTA, 2005; HOSTINS; RAABE, 2007; PÍCCOLO et al., 2010). Ao depurar código o aluno se atenta para o comportamento do algoritmo e pode expressar uma hipótese de solução, testá-la e depurá-la. De forma parecida, haja vista que há melhor representação visual dos dados, a animação de algoritmos permite ao aluno visualizar graficamente as operações do programa em execução, ao invés de fazê-lo apenas mentalmente; além disso, deixa claro para o programador em que parte da execução ocorrem erros. Pelos menos inicialmente é recomendável fornecer essas facilidades para diminuir as dificuldades com abstração; posteriormente, os estudantes estarão aptos a compreender sem auxílio gráfico o comportamento dos programas.

5 APLICAÇÃO DOS ARTEFATOS EDUCACIONAIS

Nesta seção, se analisa a quais problemas educacionais podem ser aplicados cada artefato educacional, levando-se em conta as dificuldades e necessidades identificadas pela pesquisa na revisão bibliográfica. Depois disso, com base na inclinação dos autores para uma iniciação mais gradual e na experiência positiva de Malan (2010) no resenho de um curso de programação, segue uma proposta da sequência de artefatos a ser empregada no ensino

5.1 VINCULAÇÃO DE ARTEFATOS EDUCACIONAIS A PROBLEMAS E NECESSIDADES

Esta seção apresenta de forma detalhada como os artefatos se vinculam aos escopos de problemas, do ponto de vista de se trabalhar as dificuldades e necessidades dos alunos. Assim, procura-se explicitar quais os objetivos da aplicação de cada artefato educacional no ensino de programação introdutória.

5.1.1 Soluções e problemas vinculados aos jogos

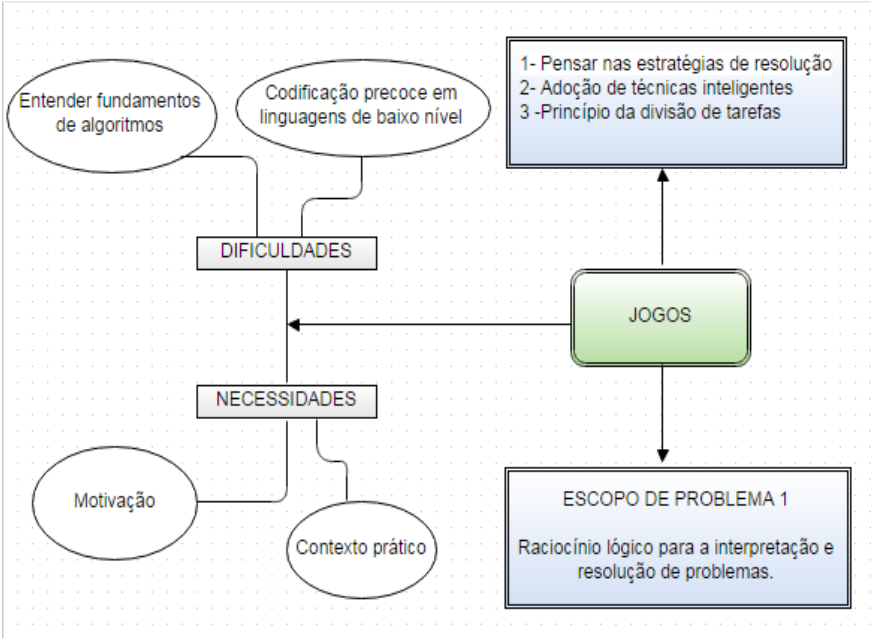
Na seção 2.1, salientou-se que a construção de algoritmos é essencialmente uma atividade de resolução de problemas. De acordo com Polya (1975) e Dante (2002), para resolver problemas é fundamental subdividi-los e abordá-los por etapas, ou seja, é preciso seguir um plano ou uma estratégia. Essa abordagem metódica é difícil de ser praticada por alunos com dificuldades no raciocínio lógico (PIVA JR; FREITAS, 2010, NETO; SCHUVARTZ, 2007, SALES; DANTAS, 2011) ou que estão acostumados a decorar conteúdos (RAPKIEWICZ et al., 2006). Por outro lado, esses mesmos alunos, segundo Hu (2008), são uma geração fascinada por jogos que, de acordo com Savi (2011) e Rapkiewicz (2006) exercitam o desenvolvimento de estratégias e a maneira certa de pensar. Dessa forma, a adoção de jogos para o ensino de programação é uma alternativa que aproveita a experiência prévia dos estudantes, trazendo um contexto prático de referência, Além desse benefício, jogos são motivadores, pois proporcionam uma aprendizagem mais divertida e prática.

Além das dificuldade com raciocínio lógico, a literatura salienta que o contato precoce com linguagens de programação de baixo nível para a construção de algoritmos pode contribuir para a desmotivação e para a perda do foco na solução dos problemas. (Melo et al., 2011, MENDES et al, 2012; NETO, 2013; RIBEIRO, 2012). De acordo com Malan (2010), durantes as primeiras semanas, a sintaxe das linguagens tende a distrair os alunos dos conceitos de programação fundamentais. Nesse sentido, Rapkiewicz et al. (2006) salientam o

uso de jogos, na fase introdutória, como uma forma de representar soluções algorítmicas sem o uso de um linguagem pré-definida.

Considerando-se, portanto, os conceitos fundamentais intrínsecos à programação, bem como as principais dificuldades e necessidades dos alunos, a presente pesquisa apresenta os jogos dentro de um esquema que deixe claro as razões para sua aplicação em um curso de programação introdutória, conforme ilustrado na Figura 2:

Figura 2 – Jogos vinculados às soluções e problemas



Fonte: elaborada pelo autor.

A Figura mostra um esquema que vincula o uso de jogos ao escopo de problema 1 (desenvolvimento do raciocínio lógico), às dificuldades de entender os fundamentos de algoritmos; aos problemas decorrentes do uso precoce de linguagens de programação; às necessidades de motivação e de contexto prático e aos conceitos e metas que se pretende atingir. Em outras palavras, a utilização de jogos para o ensino de programação visa ao desenvolvimento do raciocínio lógico, busca proporcionar um ambiente de motivação com contexto prático e divertido e significa introduzir os conceitos fundamentais de algoritmos sem a obrigação de uma linguagem de programação formal.

5.1.2 Problemas e soluções vinculados aos exercícios contextualizados

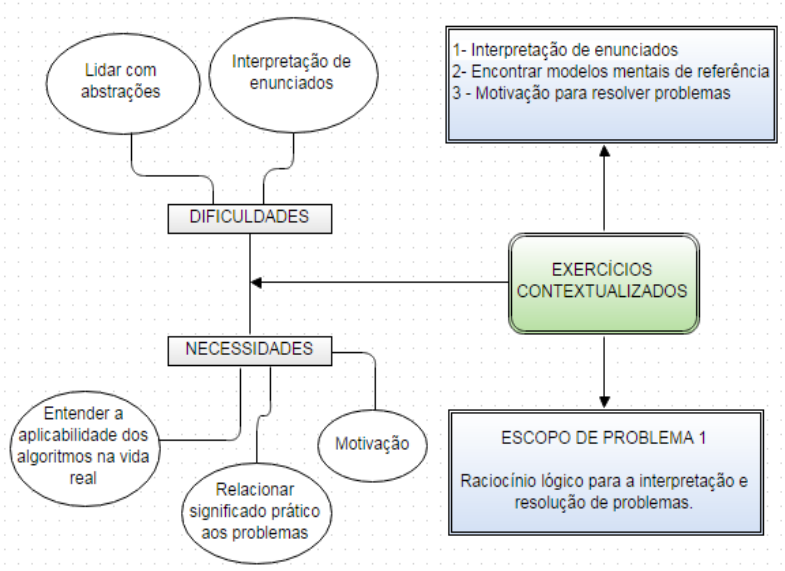
Lidar com abstrações é uma das grandes dificuldades da maior parte dos alunos no processo de ensino e aprendizagem de programação (RAPICWIEZ et al, 2006; SANTOS; SOUSA, 2005). Segundo Zanini (2012), é comum a abordagem de proposições de problemas

por meio de enunciados em lista de exercícios. Essa forma de apresentar problemas, quando os mesmos se restringem a " calcule isso" e "resolva aquilo", envolve um alto nível de abstração, além disso não deixa claro a importância do conteúdo, o que para Borges (2000) dificulta a motivação dos alunos. Para Jesus (2010) quanto menos contextualizado o enunciado mais abstrato ele se torna. Outro problema importante é o perfil de estudantes que chegam ao ensino superior sem um base adequada em português e matemática, o que dificulta a interpretação e resolução de problemas. (PIVA JUNIOR, 2010).

Segundo Zanini (2012 apud Ausabel, 1980), "o fator isolador que mais influencia a aprendizagem é aquilo que o aluno já sabe, o professor deve conhecer o que o aluno sabe e fazer deste a base de seu ensinamento". Seguindo essa abordagem, fornecer exercícios contextualizados é uma forma de facilitar o raciocínio lógico para a elaboração de algoritmos a partir da elucidação da significância e aplicabilidade dos problemas.

Considerando-se, então, as dificuldades dos alunos com a elaboração de soluções para os problemas, agravadas pela abordagem de ensino descontextualizada, este estudo propõe o uso de exercícios contextualizados, conforme ilustrado na Figura 3:

Figura 3 – Exercícios contextualizados



Fonte: elaborada pelo autor.

A Figura mostra de forma sintética a ligação dos exercícios contextualizados às dificuldades dos alunos com abstrações e interpretação de enunciados, das quais decorrem as necessidades de se fornecer significado prático aos problemas, de forma que os alunos se motivem, ao perceberem a aplicabilidade dos algoritmos na vida real.

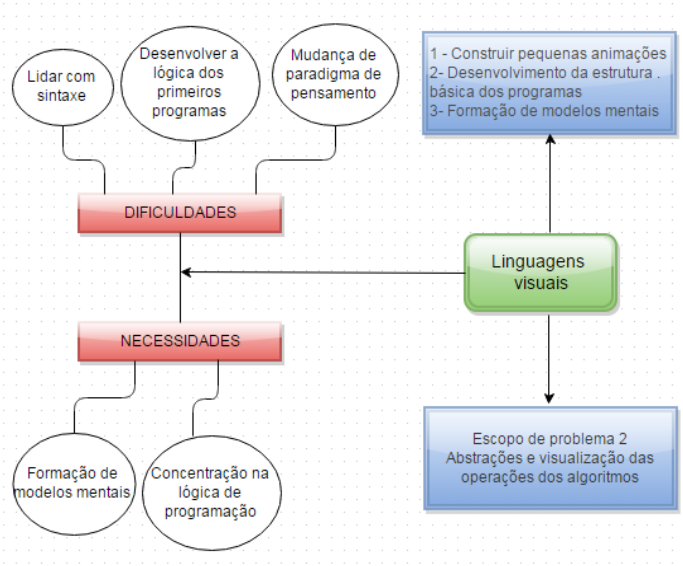
5.1.3 Problemas e soluções vinculados às linguagens visuais

De acordo com Valask (2012), a necessidade dos alunos desenvolverem raciocínio lógico, expressando-o através de uma linguagem de programação com sintaxe bastante específica, é determinante para o alto índice de reprovações em disciplinas ligadas a programação de computadores. Para Melo (2011, p.1) "a complexidade dos ambientes de desenvolvimento são, frequentemente, obstáculos ao aprendizado".

Por causa das dificuldades dos alunos em lidar com a sintaxe das linguagens tradicionais, linguagens visuais orientadas para a construção de algoritmos através do encaixe de blocos de comandos estão sendo utilizadas com sucesso para introduzir os primeiros conceitos de programação, tais como declarações, expressões booleanas, condições, loops e variáveis (MALAN; 2010; RIBEIRO et al., 2012). O intuito dessas linguagens didáticas é diminuir preocupações com sintaxe, de modo que os estudantes se foquem na lógica de resolução dos problemas. Além disso, conforme Melo (2011), os blocos de comandos de linguagens visuais guardam similaridade com as formas tradicionais de estruturas de algoritmos, permitindo que os estudantes usem a codificação em linguagem de alto nível como modelo para a construção de códigos em linguagens de programação profissionais.

A Figura sintetiza como linguagens visuais podem ser aplicadas no ensino de programação introdutória, conforme ilustrado na Figura 4:

Figura 4 – Linguagens visuais



Fonte: elaborada pelo autor.

Segundo a imagem, linguagens visuais são apontadas por este estudo, levando em conta as dificuldades dos estudantes iniciantes com a sintaxe das linguagens e com a construção dos primeiros programas. Devido a essas dificuldades, os estudos apontam para a importância de se fazer com que os alunos se concentrem na lógica de programação e também para o fato de

que é possível que os alunos formem modelos mentais a partir das linguagens visuais didáticas.

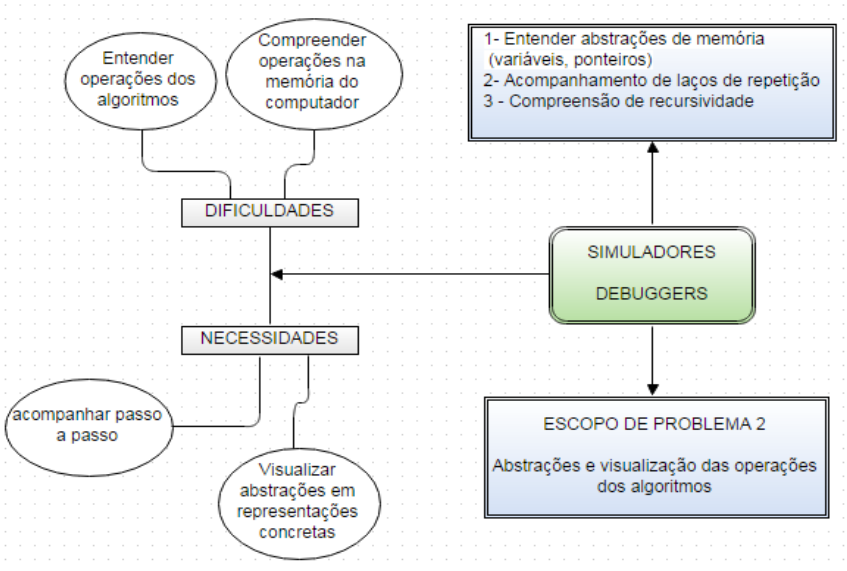
5.1.4 Problemas e soluções vinculados aos simuladores e debuggers

De acordo com Rapcwiez (2006), para a compreensão das operações realizadas pela máquina é necessário um grande esforço cognitivo do aluno para abstrair e entender como ocorrem internamente essas operações. Aureliano (2012) ressalta, por exemplo, as dificuldades dos alunos com recursão e ponteiros. Esta situação ocorre, principalmente, em aulas expositivas nas quais o professor utilizar apenas o recurso de gestos com as mãos para representar os conceitos abstratos.

Com intuito de minimizar a dificuldade do entendimento dos algoritmos, tem-se construído ferramentas de visualização que permitam simulação e/ou depuração de algoritmos (SOUZA; GIRALD, 2011; SANTOS; COSTA, 2005; HOSTINS; RAABE, 2007; PÍCCOLO et al., 2010). Segundo Soares (2004), a partir de observações em disciplinas de graduação, percebe-se um melhor resultado no aprendizado por meio de atividades práticas que façam uso de ferramentas visuais didáticas de representação de conceitos abstratos.

No caso específico deste estudo, os simuladores e debugger são as ferramentas de visualização consideradas para tratar dos problemas relacionados ao escopo de problema 2, como ilustrado na Figura 5.

Figura 5 – Simuladores e Debuggers



Fonte: elaborada pelo autor.

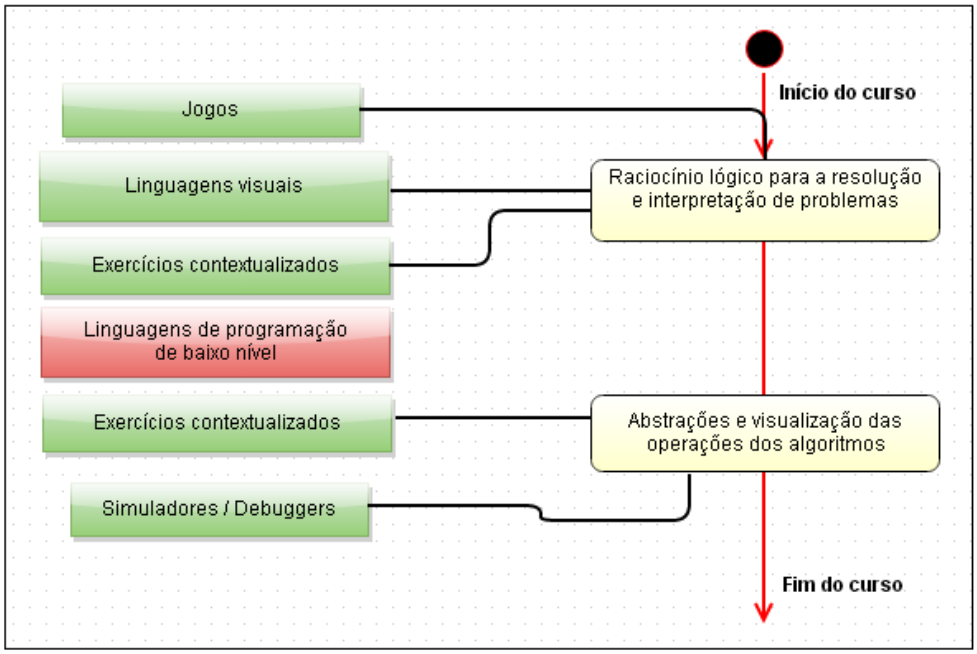
A imagem mostra a utilização de simuladores e debuggers ligada às dificuldades dos alunos em entender as abstrações da execução dos algoritmos, aos conceitos cuja

assimilação se pretende facilitar, e às necessidades decorrentes da complexidade do processo de ensino e aprendizagem de algoritmos.

5.2 SEQUÊNCIA DE UTILIZAÇÃO DOS ARTEFATOS EDUCACIONAIS

Tomando como base o programa de conteúdo do curso semestral de programação introdutória considerado, e a análise da revisão bibliográfica realizada, que aponta para uma inserção à programação de forma mais gradual, este estudo propõe uma ordem de utilização dos artefatos educacionais. Essa ordem de utilização, cuja fundamentação será descrita abaixo, é ilustrada na Figura 6

Figura 6 – Sequência de utilização dos artefatos educacionais



Fonte: elaborada pelo autor.

No lado direito da Figura, observam-se um círculo negro, que indica o início de um curso introdutório de programação, e uma linha representando temporalmente as fases desse curso. Interceptando a linha que parte do circulo negro, notam-se duas caixas brancas: a primeira se aplica à parte inicial do curso, e significa a utilização de jogos, linguagens de programação visuais e exercícios contextualizados; a segunda se aplica à parte intermediária do curso, estendendo-se até a final, e significa o uso de exercícios contextualizados, *debuggers* e simuladores.

A ordem de uso dos artefatos educacionais ilustrada na figura 02 se baseia, em boa parte, no curso CS50 da universidade de Harvard, que introduz os fundamentos da ciência da computação e a arte de programação. Considerando que em 1996 o número de matriculados no CS50 era de 386 e, em 2006, de 132, Malan (2010) relata os possíveis erros e

as mudanças que foram realizadas a partir de 2007 para aumentar o número de matrículas no curso. Dentre os possíveis equívocos, o autor menciona o fato de o curso ter adotado nos últimos 10 anos uma abordagem inicial de muito baixo nível, introduzindo primeiramente à linguagem *assembly* e depois C. Além disso, destaca que o tradicional programa "*Hello World*" tende a não despertar o interesse de alunos que carregam *IPhones* e *IPods*, e que, nas primeiras semanas, a sintaxe da maioria das linguagens distrai os estudantes dos conceitos fundamentais mais importantes.

Depois de analisar a antiga forma de ensino do curso CS50, uma das medidas tomadas por Malan (2010) e sua equipe foi a de modernizar os conjuntos de problemas, amarrando-os para um domínio do mundo real (**contextualização**) mais familiar para os estudantes. Outra medida tomada por eles consistiu na utilização da linguagem Scratch para dois objetivos principais: o primeiro foi o de fazer com que os alunos construíssem seus primeiros jogos e animações, sem preocupação com vírgulas e com outras distrações sintáticas que não são interessantes nos primeiros dias de curso; o segundo foi o de introduzir os conceitos de declarações, expressões booleanas, condições, loops e variáveis. Essas ações de redesenho resultaram em um aumento de 156% do número de inscritos entre 2007 e 2009, e podem culminar em pelo menos três consequências: produzir uma **boa carga de motivação** nos alunos, direcionar os estudantes para os **conceitos fundamentais de algoritmos** e **permitir a transição dos conteúdos mais simples para os mais complexos de forma mais gradual**. Baseado nestas três consequências e nas recomendações por trás do uso dos artefatos educacionais, a ordem de utilização dos artefatos propostos por esta pesquisa visa alcançar cinco objetivos:

1. Nas primeiras semanas, aplicar uma grande carga de motivação nos alunos;
2. Mostrar a importância e aplicabilidade da computação no dia a dia;
3. Desenvolver os primeiros passos da lógica de programação com o uso de jogos e linguagens de programação visuais;
4. Tornar mais gradual a transição dos conteúdos mais simples para os mais complexos;
5. Apresentar conceitos abstratos por meio de representações mais concretas.

Como visto na seção 4.1.1, para atingir o primeiro e o terceiro objetivos, o uso de jogos é amplamente indicado e divide-se em duas vertentes: a primeira consiste na programação dos próprios jogos (MALAN, 2010, MARQUES et. al, 2010), e a segunda representa resolver problemas algoritmicamente no próprio ato de jogar. Vale ressaltar aqui a segunda vertente de uso dos jogos, que permitir aos alunos iniciar os fundamentos de

programação sem a utilização de uma linguagem de programação propriamente dita. Para alcançar o segundo objetivo, são indicados exercícios contextualizados que representam situações do dia a dia, bem como o empenho dos professores em indicar a presença da computação no cotidiano com exemplos, vídeos e comparações.

Utilizar jogos e linguagens de programação visuais no início do curso é uma forma de criar modelos mentais de algoritmos que sirvam de base para o aprendizado de programação em linguagens de uso comercial. Dessa maneira, é possível comparar, por exemplo, as construções dos blocos de comandos de uma linguagem mais simples com as de uma linguagem mais complexa, facilitando desse modo a compreensão. Com isso, procura-se atingir o quarto objetivo: tornar mais suave a transição de uma etapa de conteúdo para outra. Finalmente, uma das maneiras de alcançar o quinto objetivo é a utilização de simuladores e *debuggers* para facilitar a visualização das operações dos algoritmos.

6 BUSCA E SELEÇÃO DOS ARTEFATOS EDUCACIONAIS

A maior parte dos artefatos educacionais foi encontrada por meio dos mecanismos de busca da internet com os seguintes termos: *animate algorithms, animate recursive algorithms*, ferramentas *debugger*. As buscas foram realizadas com termos em inglês, porque houve pouquíssimo retorno relevante em português. Algumas ferramentas apresentadas nos simpósios e workshops nacionais de informática na educação foram adotas; os exercícios contextualizados foram selecionados do site SPOJ* a partir de indicação Tavares (2012).

6.1 SELEÇÃO DOS JOGOS PARA ENSINO DE PROGRAMAÇÃO

A revisão bibliográfica sobre o ensino de programação com jogos revelou duas vertentes: a primeira consiste na implementação de jogos em linguagens visuais, e a segunda, **adotada nessa pesquisa**, consiste no ensino de programação no ato de jogar, em outras palavras, o aluno aprende os conceitos jogando.

Os critérios de qualidade para jogos educacionais relatados na seção 2.5 foram utilizados para a escolha dos jogos que, segundo as premissas do trabalho, se encaixam melhor em um curso de programação introdutória; isso, porém, não quer dizer que os não selecionados tenham sido classificados como inutilizáveis, pois todos apresentaram pontos positivos a serem explorados. Em linhas gerais, buscou-se selecionar jogos pragmáticos, com

* br.spoj.com

objetivos pedagógicos bem definidos (aderentes ao conteúdo de programação introdutória), fáceis de usar e com riqueza gráfica razoável. Esses critérios foram resumidos em sete requisitos, de acordo com (SAVI; 2011; MORATORI; 2013):

1. Um jogo deve, portanto, fornecer um mínimo de riqueza gráfica e interatividade que não estejam tão distantes dos softwares comerciais;
2. É recomendável que o jogo educacional forneça evidências claras dos conceitos que se propõe a ensinar e das habilidades que visa desenvolver (objetivos bem definidos);
3. Possibilidade de formação de conceitos;
4. Execução rápida e sem erros;
5. Acesso à ajuda;
6. Possibilidade de controle do usuário sobre a sequência de execução do sistema;
7. Possibilidade de correção de respostas;

6.1.1 Jogos examinados que não foram selecionados

Nesta seção, são apresentados como exemplos dois jogos que foram verificados, de acordo com os critérios de qualidade definidos na seção 6.1, mas que não foram admitidos para compor a solução proposta pelo trabalho; mesmo assim, são artefatos educacionais utilizáveis que servem como referência qualitativa para o uso de jogos, de tal forma que se tenha uma visão dos pontos negativos e positivos.

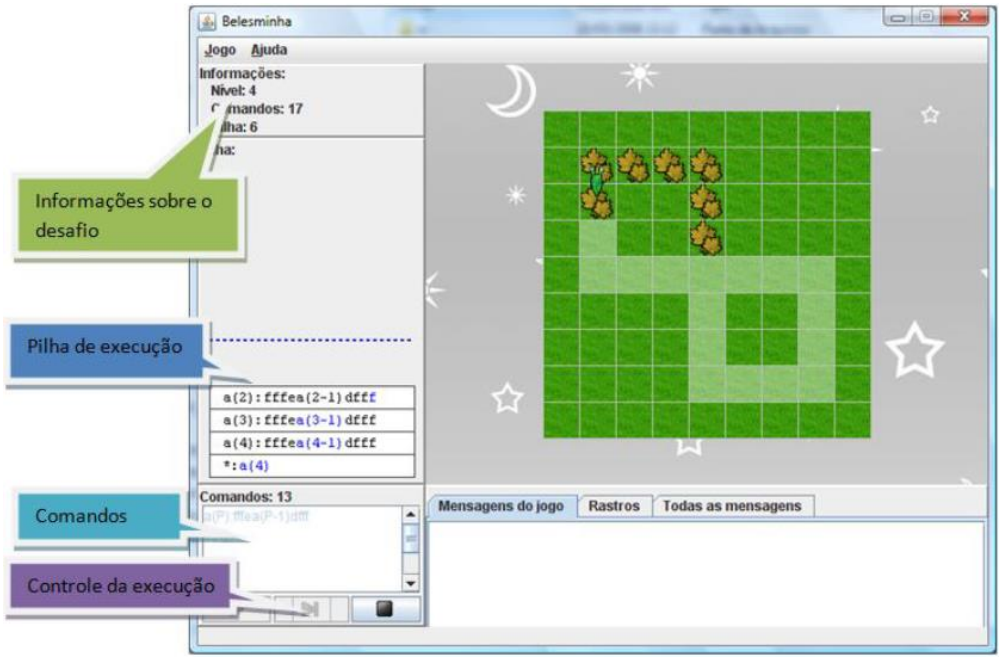
6.1.1.1 Belesminha

É um jogo para auxiliar na aprendizagem dos conceitos de recursividade e pilha de execução. Consiste em fazer com que uma lesma – personagem Belesminha – percorra uma trajetória para recolher folha de laranjeiras espalhadas pelo quadrante. Três comandos determinam os movimentos do personagem: ir para frente, vira 90º à direita, virar 90º à esquerda. Por ter objetivos bem definidos (**Requisito 2**), por mostrar de forma simples a pilha de execução (**Requisito 3**), pela evolução gradual do nível de seu nível de dificuldade, pela execução rápida e sem erros (**Requisito 4**), Belesminha é um jogo com bastante potencial para auxiliar no aprendizado de recursividade.

Há, porém, três qualidades importantes que ainda não foram satisfeitas integralmente no jogo: a primeira é a qualidade gráfica (**Requisito 1**), a segunda é o acesso a ajuda (**Requisito 5**), que está presente em uma página web, e não internamente, e a terceira é a possibilidade de controle do usuário sobre a sequência de execução do sistema (**Requisito**

6), já que o sistema não permite ao usuário escolher uma fase. Por causa desses pontos a desenvolver, Belesminha ficou com classificável, podendo ser utilizado com as devidas orientações e cuidados.

Figura 7 – Jogo Belesminha



Fonte: Coutinho et al (2008)

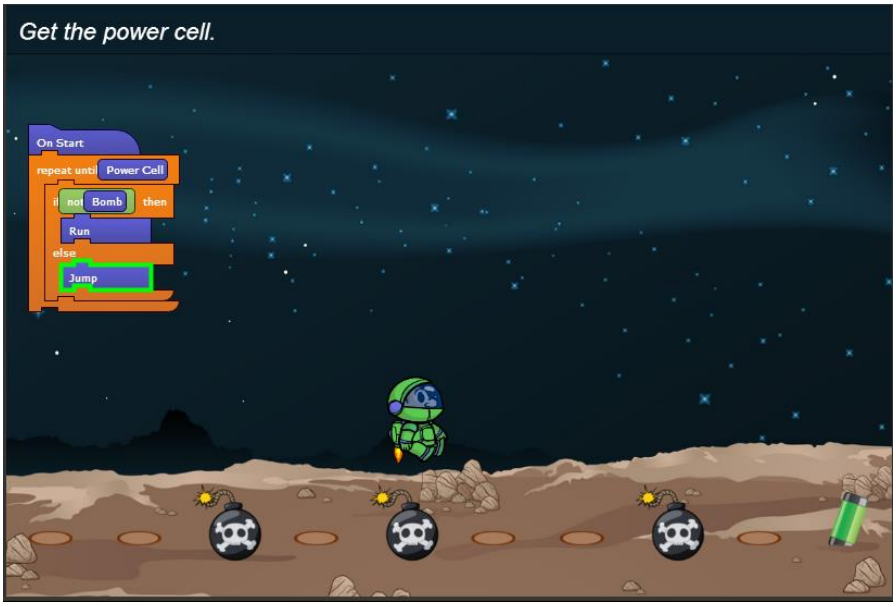
6.1.1.2 Lost in space

Lost in Space é um jogo online da plataforma Tynker* que consiste em fazer com que um astronauta capture os objetos indicados. Os movimentos do personagem devem ser programados para que ele desvie dos obstáculos ou siga o caminho correto até um objeto. Por meio de programação visual, implementada pelo arrastar de blocos de comando, o aprendiz vai paulatinamente sendo introduzido na arte de programar, absorvendo de forma mais intuitiva os conceitos de laços de repetição e condicionais

Com gráficos nos moldes do famoso desenho Bob Esponja, o jogo tem uma rica interface gráfica (**Requisito 1**) voltada para o público infantil, pois a platarforma Tynker constitui-se de um conjunto de jogos para ensinar crianças a programar; mesmo assim, não deixa de ser um ótimo recurso introdutório para adolescentes e jovens, pois os resultados da execução dos blocos de comandos são claramente expressos em eventos gráficos. A figura 8 ilustra uma das fases do jogo:

* www.tynker.com

Figura 8 – Jogo Lost in Space



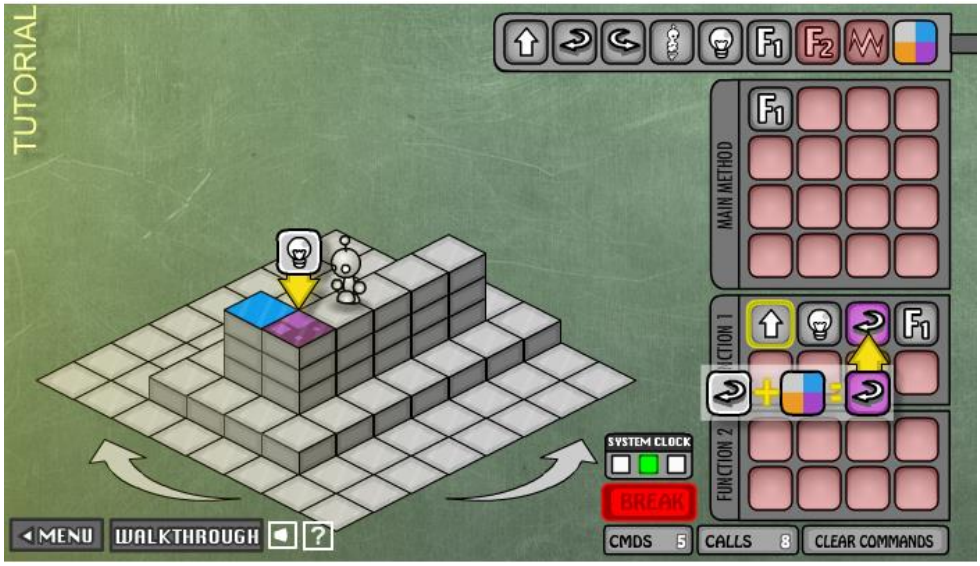
Fonte: Plataforma Tynker

A forma como o jogo apresenta os blocos de comandos diminui o nível de abstração, tornando os resultados do algoritmo visualmente concretos; assim tem-se a possibilidade de formação de conceitos (**Requisitos 2 e 3**). Á medida que o usuário erra o algoritmo, o sistema mostra dicas para conduzir o jogador até a resposta correta; basta, então, recombinar os blocos de comando para formar o novo algoritmo (**Requisitos 5 e 7**). Apesar de todas essas qualidades, o jogo não foi selecionado, pois tem uma execução bastante lenta. Talvez isso seja uma característica que leva em conta o público infantil.

6.1.2 Jogos selecionado: light bot2

Ligth Bot 2 é um jogo de trajetória, cuja tarefa consiste em fazer com que um robô acenda as luzes de determinadas quadrículas localizadas num ambiente tridimensional. A trajetória do robô e o acendimento das luzes são realizados a partir dos comandos: para frente, virar à direita, virar à esquerda, pular, acender, função1, função2 e *break*. A análise do jogo indicou que a construção das trajetórias e o aninhamento dos comandos são capazes de transmitir os conceitos fundamentais de algoritmos (Requisito 3): modularização, comandos de decisão e repetição, funções, recursividade e estrutura básica de programas. A figura 09 ilustra o jogo e evidencia todos os seus comandos:

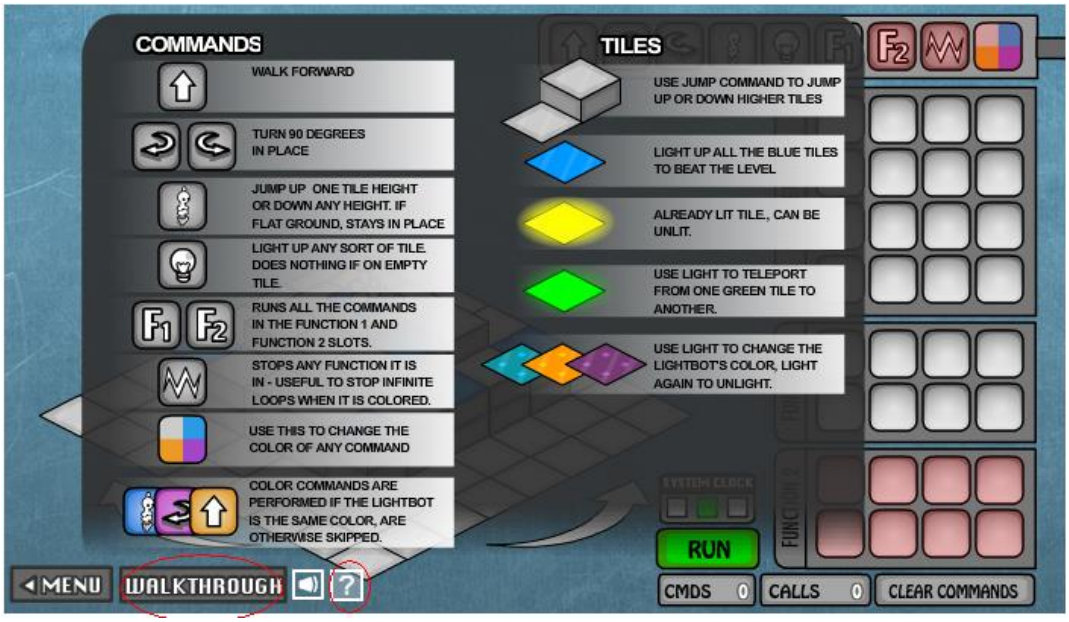
Figura 9 – Tela do Jogo Light Bot2



Fonte: Elaborado pelo autor

Na parte superior da figura 09, estão localizados os comandos, representados por símbolos, que devem ser arrastados para as quadrículas, o que torna simples e intuitivo a interação com o jogo (Requisito 1). Logo abaixo, segue o *main method* (método principal*)* e, em seguida, *a function1 e function 2*; observa-se também *system clock*, que controla a velocidade de execução (Requisitos 6,7), e *break*, que para a execução, sendo substituída em seguida pelo comando *run*, que aciona os comandos. Na parte inferior esquerda, o jogo fornece duas excelentes opções de ajuda que podem ser conferidas na figura 10:

Figura 10 – Ajuda Light Bot2



Fonte: Elaborado pelo autor

Ao se colocar o ponteiro do mouse no símbolo de interrogação, parte inferior da figura 10, surge uma janela com a descrição do significado de cada símbolo de comando (Requisito 5). Clicando-se em *walkthrough*, abre-se uma janela com vídeos exemplos do jogo. Como numa linguagem de programação profissional, o jogo possui um conjunto de comandos que são chamados a partir do método principal. Analisando a figura é possível evidenciar os seguintes conceitos: estrutura básica de um programa (método principal que inicia os comandos e chama as funções); modularização por meio dos comandos dentro das funções (dividir para conquistar); comandos de decisão, pois o robô só vira para a esquerda se o presente quadradinho que pisa for da cor roxa; recursividade a partir da chamada ao comando F1 na função1; laço de repetição implementado pela recursividade.

6.2 SELEÇÃO DOS EXERCÍCIOS CONTEXTUALIZADOS

Na seção 4.1.2, citou-se a pesquisa de Zanine e Raabe (2012), que coletou 428 enunciados dos livros adotados nas disciplinas de programação introdutória dos cursos de T.I e ciência da computação no Brasil. Nessa coleta, 64,72% dos exercícios fazem parte da categoria de problemas descontextualizados. Com base nos resultados destes pesquisadores, ficou claro que procurar exercícios contextualizados nas bibliotecas dos cursos de computação não era uma boa alternativa.

Como opção, a pesquisa aproveitou o trabalho realizado por Tavares (2012), na Universidade Federal do Ceará Campus Quixadá, como tutor dos grupos de estudos para a OBI (Olimpíadas Brasileiras de Informática). O professor selecionou um conjunto de exercícios do SPOJ, ferramenta online para treinamento em algoritmos, e o subdividiu em grupos de acordo com os conceitos abordados, tais como: leitura e saída, comandos de seleção e repetição, operador módulo, vetores, matrizes, recursão e strings. Este conjunto de problemas foi selecionado para compor os artefatos educacionais resultantes da pesquisa.

Os exercícios selecionados variam de simples implementações para algoritmos mais complexos, o que permite adequar os exercícios ao nível de cada aluno; assim, o normal é passar em sala de aula exercícios simples, mas também é interessante deixar alguns problemas desafiadores para coroar a disposição daqueles que querem ir além da média. Foram coletados 50 exercícios para serem utilizados como uma alternativa de substituição de boa parte dos exercícios descontextualizados por problemas de programação que estejam claramente ligados às situações do mundo real.

Um dos 50 problemas, ilustrado no Quadro 5, cita uma brincadeira comum na infância para inserir o algoritmo em uma situação cotidiana. A solução da questão, por ser

razoavelmente simples, é uma demonstração de que os exercícios selecionados podem ser aplicados em turmas de programação introdutória.

Quadro 5 – Problema Bafo

Problema: BAFO

Álbuns de figurinhas – sejam de times de futebol, princesas ou super-heróis – têm marcado gerações de crianças e adolescentes. Conseguir completar um álbum é uma tarefa muitas vezes árdua, envolvendo negociações com colegas para a troca de figurinhas. Mas a existência das figurinhas propicia uma outra brincadeira, que foi muito popular entre crianças no século passado: o jogo de bater figurinhas (o famoso "Bafo"). O jogo é muito simples, mas divertido (e muito competitivo). No início de uma partida, cada criança coloca em uma pilha um certo número de figurinhas. Uma partida é composta de rodadas; a cada rodada as crianças batem com a mão sobre a pilha de figurinhas, tentando virá-las com o vácuo formado pelo movimento da mão. As crianças jogam em turnos, até que a pilha de figurinhas esteja vazia. Ganha a partida a criança que conseguir virar mais figurinhas.
Aldo e Beto estão jogando bafo com todas as suas figurinhas e pediram sua ajuda para calcular quem é o vencedor.

Tarefa

Você deve escrever um programa que, dada a quantidade de figurinhas que Aldo e Beto viraram em cada rodada, determine qual dos dois é o vencedor.

Entrada

A entrada é composta de vários casos de teste, cada um correspondendo a uma partida entre Aldo e Beto. A primeira linha de um caso de teste contém um número inteiro R que indica quantas rodadas ocorreram na partida. Cada uma das R linhas seguintes contém dois inteiros, A e B, que correspondem, respectivamente, ao número de figurinhas que Aldo e Beto conseguiram virar naquela rodada. Em todos os casos de teste há um único vencedor (ou seja, não ocorre empate). O final da entrada é indicado R=0

Saída

Para cada caso de teste da entrada, seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do caso de teste, no formato "Teste n", onde n é numerado seqüencialmente a partir de 1. A segunda linha deve conter o nome do vencedor (Aldo ou Beto). A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo Entrada:	Saída:
2	Teste 1
1 5	Beto
2 3	
0	

Fonte: br.com.spoj

Para resolver este problema, o aluno só precisa fazer com que duas variáveis acumulem o número de figurinhas levantadas para Aldo e Beto em cada rodada; no final das rodadas, basta comparar qual das duas variáveis é maior para determinar o vencedor, como ilustrado no código da Figura 1

Figura 11 – código de resolução do exercício

```
#include <stdio.h>
int main()
{
   int r,i,totalAldo,totalBeto, figurasAldo, figurasBeto;
   int teste = 1;
    while(1)
    {       scanf("%d",&r);
            if(r==0) return 0;

            totalAldo=0;  totalBeto=0;
            for(i=0; i<r; i++)
            {
            scanf("%d %d",&figurasAldo,&figurasBeto);
            totalAldo+=figurasAldo;  totalBeto+=figurasBeto;
                                    }
            printf("Teste %d\n", teste);
            if(totalAldo>totalBeto)
            printf("Aldo\n\n");
            else
            printf("Beto\n\n");

            teste++;
                    }
}
```

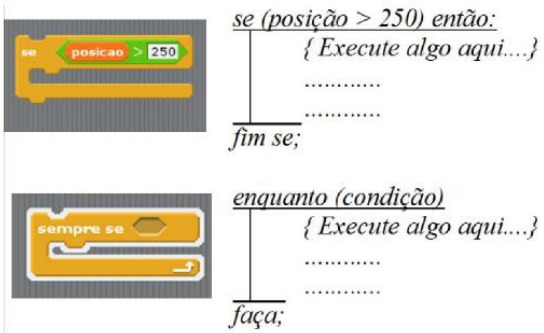
Fonte: elaborado pelo autor

O código do exemplo foi submetido ao corretor do site e aceito como resposta correta. Este é um excelente recurso para conferir a corretude dos algoritmos, os quais passam por um extenso e rigoroso conjunto de testes. A lista com os demais exercícios se encontra no Apêndice A.

6.3 SELEÇÃO DA LINGUAGEM DE ALTO NÍVEL SCRACTH

Scratch é uma linguagem visual criada para introduzir de forma intuitiva os principais fundamentos de programação, tais como a estrutura básica de programas, comandos de decisão, laços de repetição e quebra dos problemas em parte menores. Um característica importante, observada por Melo (2011), é que os blocos do Scratch guardam similaridade com as formas tradicionais dos algoritmos codificados em linguagens profissionais, como ilustrado na Figura 12:

Figura 12 – Blocos de comandos de Scracth



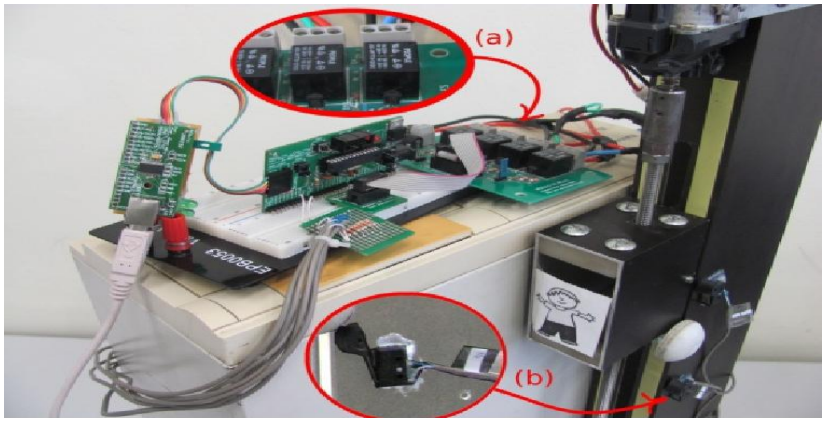
Fonte: Melo (2011)

É essa similaridade que permite a criação de modelo mentais de referência para a codificação de algoritmos nas principais linguagens de programação.

De acordo com Mendes et al. (2012, p.2), “o uso da linguagem Scratch na introdução de conceitos algorítmicos tem sido bem vista. Segundo Santos (2013), essa linguagem produz resultados mais rápidos, evita desgaste dos alunos com sintaxe e fornece um ambiente de desenvolvimento atrativo e fácil de usar. Há várias publicações com os resultados da aplicação dessa linguagem para o ensino introdutório de lógica de programação (MELO et al, 2011; SCAICO et al, 2012; SANTOS NETO, 2013; SANTOS 2013).

Melo et al. (2011) destaca resultados positivos em experiências envolvendo Scracth e Arduindo, que é uma plataforma de prototipação baseada em hardware e software. No projeto piloto proposto pelo autor, os estudantes utilizam os recursos de programação visual de Scracth for Arduindo (versão adaptada) para controlar dispositivos eletrônicos, tais como sensores e elevadores, como ilustrado na figura 13:

Figura 13 – Scracth e Arduino



Fonte: Melo (2011)

Essa abordagem prática e contextualizada trouxe uma maior motivação aos alunos, permitindo-lhes construir aplicativos interessantes como tocadores de mp3, automatismos simples, jogos elementares, e uma casa inteligente controlada por PC (MELO et al., 2011).

Mendes et al. (2012) também relata bons resultados da utilização de Scratch em um curso conduzido com 32 alunos de turmas de 2º e 3º ano do nível médio. Com o objetivo de apresentar os principais conceitos de programação, foi ministrado curso de 20 horas e organizado uma prova aos moldes de uma olimpíada de programação. Ao final do curso, além do excelente desempenho nas provas, os alunos relataram que se sentiam motivados a continuar aprendendo outras linguagens de programação.

O estudo em questão, portanto, selecionou a linguagem de alto nível Scratch, levando em consideração os relatos dos bons resultados de sua utilização; pesou ainda o fato

de ser um ambiente de desenvolvimento que já vem sendo utilizado por alguns professores do curso de Sistemas de Informação da UFC.

6.4 SELEÇÃO DOS SIMULADORES

Conforme visto na seção 4.2.2, os simuladores devem mostrar os conceitos abstratos da execução dos algoritmos através de representações visuais concretas. De acordo com os critérios discutidos na seção 2.5.1, antes e durante a execução das simulações, as ferramentas também devem garantir requisitos de qualidade que as tornem fáceis de usar, de tal forma que o usuário se detenha minimamente com dúvidas sobre a operação.

A busca, portanto, foi realizada de acordo com três critérios principais: I - facilidade de uso, II - controle do usuário sobre a execução do sistema, que permite ao aluno controlar a velocidade e os passos da execução, III - qualidade da elucidação do algoritmo simulado, que significa o grau de clareza com que os elementos visuais são correlacionados aos algoritmos. Encontrou-se, em sua maioria, simuladores para algoritmos de ordenação, o que é até coerente pelo fato desses algoritmos exercitarem conceitos importantes como laços de repetição, vetores, comandos de decisão, leitura e escrita de variáveis. Por outro lado, para garantir uma maior cobertura do conteúdo de programação introdutória, a pesquisa encontrou um ambiente online com mais de 50 exercícios de simulação sobre algoritmos básicos e estruturas de dados; trata-se de TKALAK2 que será descrito na seção 6.4.2.3.

6.4.1 Simuladores examinados que não foram selecionados

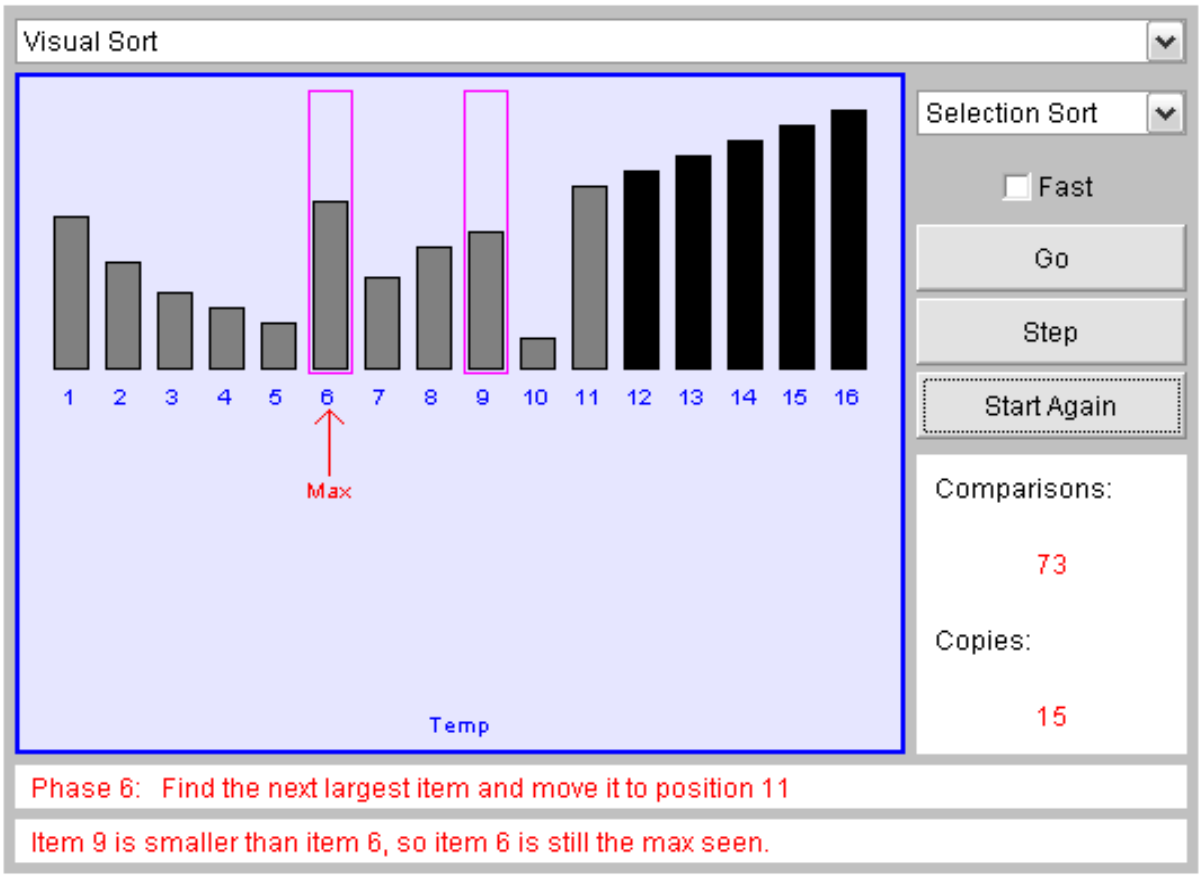
Nesta seção são apresentados como exemplos dois simuladores que, de acordo com os critérios definidos na seção 6.4, ficaram de fora da seleção. A descrição desses artefatos educacionais é feita de modo a destacar os pontos positivos, bem com os pontos negativos que os levaram a ser descartados da seleção.

6.4.1.1 SortLab: simulador de algoritmos de ordenação

SortLab é um applet Java para simulação dos principais algoritmos de ordenação. O simulador mostra através de uma representação de barras verticais como o algoritmo troca as posições dos elementos para ordená-los. Para simular uma ordenação basta selecionar no combobox um dos cinco algoritmos de ordenação (*Selection Sort*, *Bubble Sort*, *Insertion Sort*, *Quick Sort e Merge Sort*) e clicar no botão Go ou interagir passo a passo, clicando no botão

Step de forma que a interação só prossiga a cada clique. Á medida que o algoritmo prossegue, as instruções são explicados textualmente na parte inferior da figura 14:

Figura 14 – Tela de SortLab



Fonte: Elaborado pelo autor

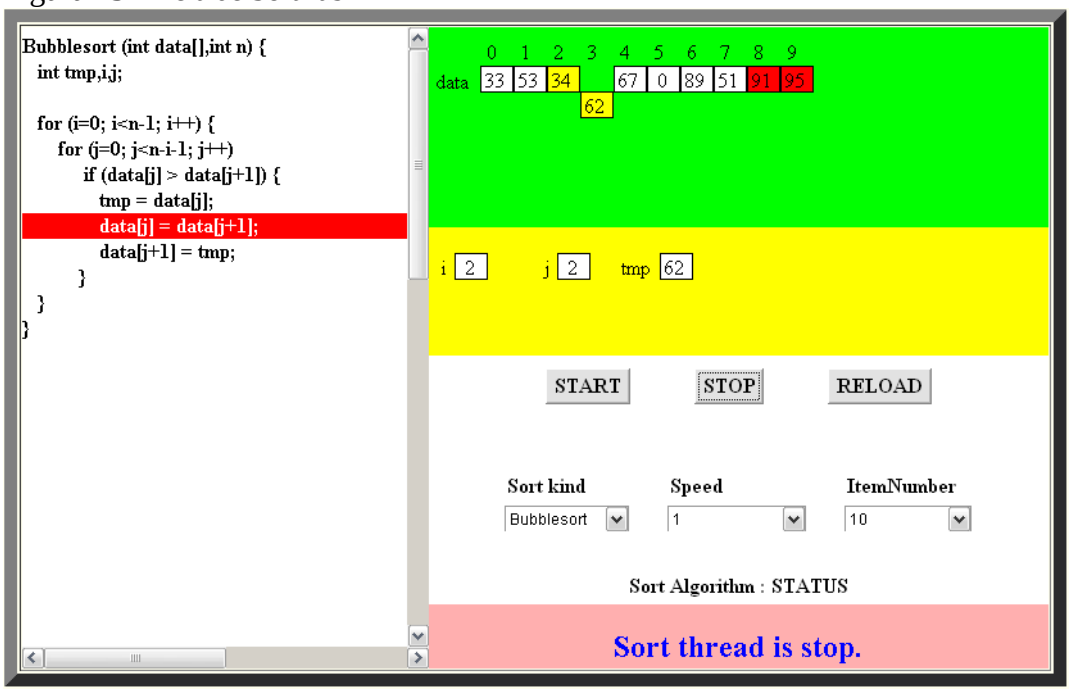
Com o mínimo de recursos necessários, SortLaB é um applet de **interface simples** e **fácil de usar**; porém, fornece para o estudante pouco **controle sobre a execução** do **algoritmo**, permitindo apenas avançar. Outro requisito importante que o simulador deixa a desejar é a **qualidade da elucidação do algoritmo**, pois não mostra nenhuma representação de linhas de código em execução e conta apenas com uma explicação textual na parte inferior. Por essas razões, Sortlab ficou de fora da seleção, mas é uma alternativa utilizável, apesar de duas limitações.

6.4.1.2 SortLab2: simulador de algoritmos de ordenação

SortLab2 é mais um dos applet Java para simulação dos principais algoritmos de ordenação. O simulador além de mostrar uma animação de caixinhas trocando de posição,

apresenta os códigos dos algoritmos sendo executados linha por linha, o que significa **um nível a mais de elucidação**. A figura 15 ilustra o programa

Figura 15 – Tela de SortLab2



Fonte: Elaborado pelo autor

A parte amarela da figura 15 contém caixinhas para mostrar a mudança dos valores das variáveis de controle, à medida que a execução avança. Isso é mais um elemento de clareza importante para o estudante. Logo abaixo da faixa amarela, encontram-se três caixas de seleção para a escolha do algoritmo a ser simulado, da **velocidade de execução e do número de elementos do vetor** (**Requisito II**). Depois de escolher as opções, basta clicar no botão *start* para iniciar a simulação.

Sortlab2, entretanto, não possui um recurso direto para a simulação passo a passo. Para implementar este recurso, o estudante precisa alternar cliques entre os botões *start* e *stop*. Por este motivo o simulador fica como alternativa secundária, em relação aos simuladores que serão apresentados na seção 6.4.2.

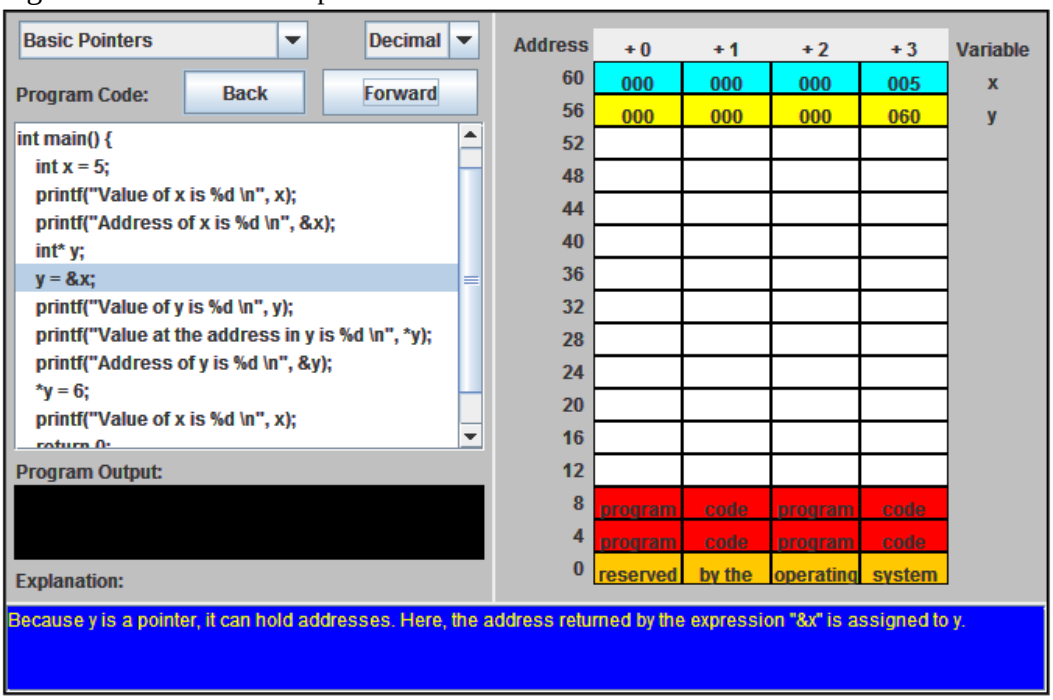
6.4.2 Simuladores selecionados

Nesta seção, são apresentados os simuladores que, segundo os critérios definidos na pesquisa, reuniram as melhores condições para serem utilizados em um curso de programação introdutória.

6.4.2.1 Simulador de ponteiros em c

Applet Java para a simulação do funcionamento de ponteiros na linguagem C. É um simulador com interface clara, simples e intuitiva (Requisito I) que auxilia o aluno a compreender as principais operações com ponteiros tais como: atribuição de endereço de variáveis primitivas, aritmética de ponteiros, uso das funções malloc(), realloc(), free(), ponteiro para ponteiro. A figura 16 ilustra a interface do simulador:

Figura 16 – Simulador de ponteiro em C



Fonte: Elaborado pelo autor

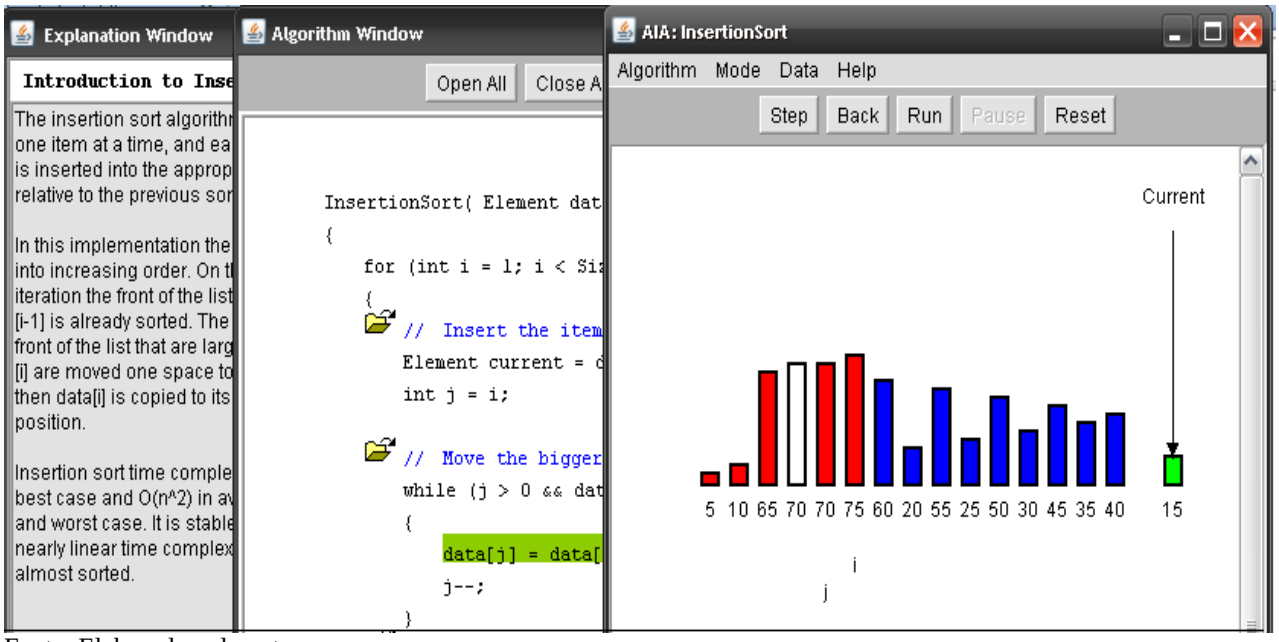
A operação é simples. Na parte superior esquerda do applet, o usuário seleciona um programa no combobox e depois clica seguidamente no botão forwad para a execução passo a passo ou em back para retroceder para os passos anteriores (Requisito II); à medida que vai clicando nos botões, a representação visual da memória é modificada e, na parte inferior do simulador, são exibidas as saídas e uma explicação didática da operação que ocorre em cada linha (Requisito III).

6.4.2.2 Sort: simulador de algoritmos de ordenação

Sort possui todos os recursos de SortLab2 descrito na seção 6.4.1.2 e outros a mais que o tornam mais rico e fácil de usar (Requisito I). Permite a execução automática, passo a passo com a possibilidade de retroceder (Requisito II), e ainda apresenta texto explicativo e código do algoritmo que está sendo simulado. A representação visual do

algoritmo é com barras verticais, que mudam de cor conforme os elementos vão sendo selecionados; há também as letras i e j, representando as variáveis de controle do algoritmo, cujas instruções são grifadas de verde, à medida que são executadas, de forma que fique evidenciado como a estrutura de dados está sendo manipulada (Requisito III). A figura 17 ilustra a interface do simulador:

Figura 17 – Tela do simulador Sort



Fonte: Elaborado pelo autor

Com todos os seus recursos, Sort pode ser visto tanto como um depurador quanto como um simulador, pois a interação mostra, ao mesmo tempo, a execução do algoritmo linha por linha, bem como o movimento dos elementos de uma posição para outra.

Visualizando a execução de programas apenas mentalmente ou acompanhando em lousa, é muito comum o aluno perder ou confundir em sua linha de raciocínio as trocas de valores das variáveis, tendo que voltar repetidamente para o início do fluxo de execução para corrigir a sequência, o que não é de todo ruim pelo processo de aprender pelo erro, porém isso exige muito mais esforço.

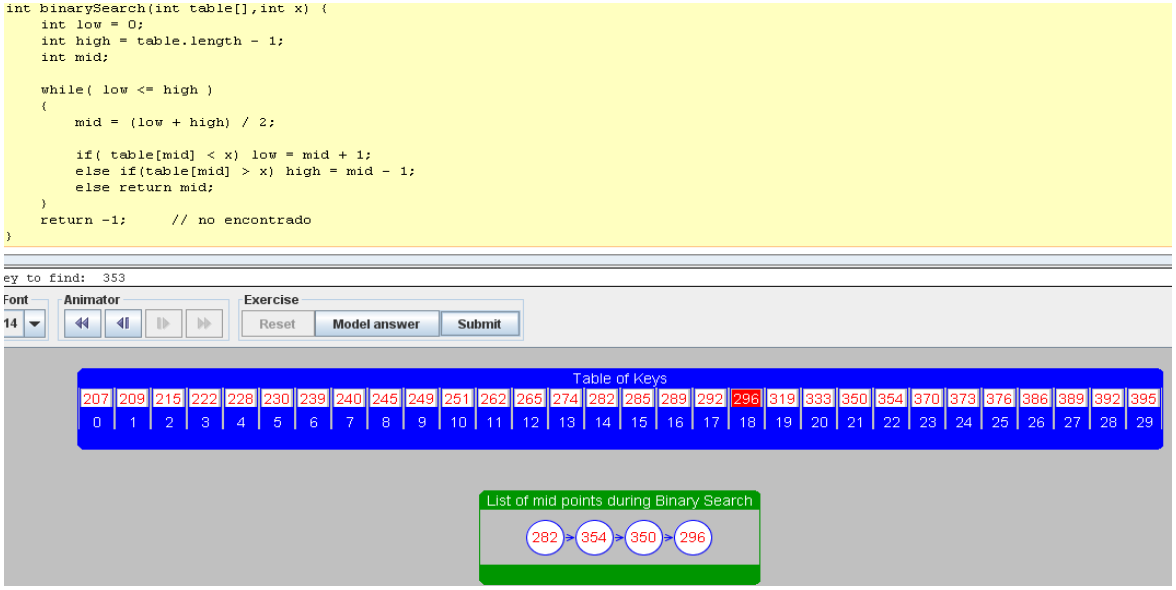
Com o objetivo de sanar estes problemas, Sort foi projetado para que o estudante não perca a sequência do algoritmo, porém, mesmo que isso aconteça, ele pode retroceder (botão Back) até o passo compreensível e seguir dali em diante. Uma técnica didática que aproveita muito o recurso passo a passo (botão Step) consiste em perguntar ou se perguntar qual será o efeito do próximo passo; dessa forma, verifica-se o nível de compreensão do algoritmo e os pontos incompreendidos. Por possibilitar toda essa facilidade de visualização,

segundo o referencial teórico na seção 4.2.2, este simulador compõe o conjunto de artefatos do resultado final da pesquisa.

6.4.2.3 Trakla2: simulador de algoritmos

Trakla2 é um ambiente online para prática de exercícios de estruturas de dados e algoritmos básicos, que fornece feedeback imediato e automático das soluções submetidas (korhonen et al., 2003). Dentre os simuladores verificados na pesquisa, é o que permite o maior nível de interação com a simulação, já que o estudante arrasta como o mouse os elementos para a representação das estruturas de dados. Dessa forma, o aluno manipula diretamente as estruturas de dados por meio de elementos gráficos com a finalidade de executar as mesmas modificações que o algoritmo realiza. O ambiente possui mais de 50 exercícios, abordando estruturas de dados básicas e os principais algoritmos usados na computação. Nas simulações de ordenação, por exemplo, o usuário seleciona um elemento em uma dada posição e o move para outra posição, podendo ainda retroceder para os passos anteriores. A figura 18 mostra como exemplo a simulação do algoritmo de busca binária:

Figura 18 – Exercício de simulação do algoritmo de busca binária

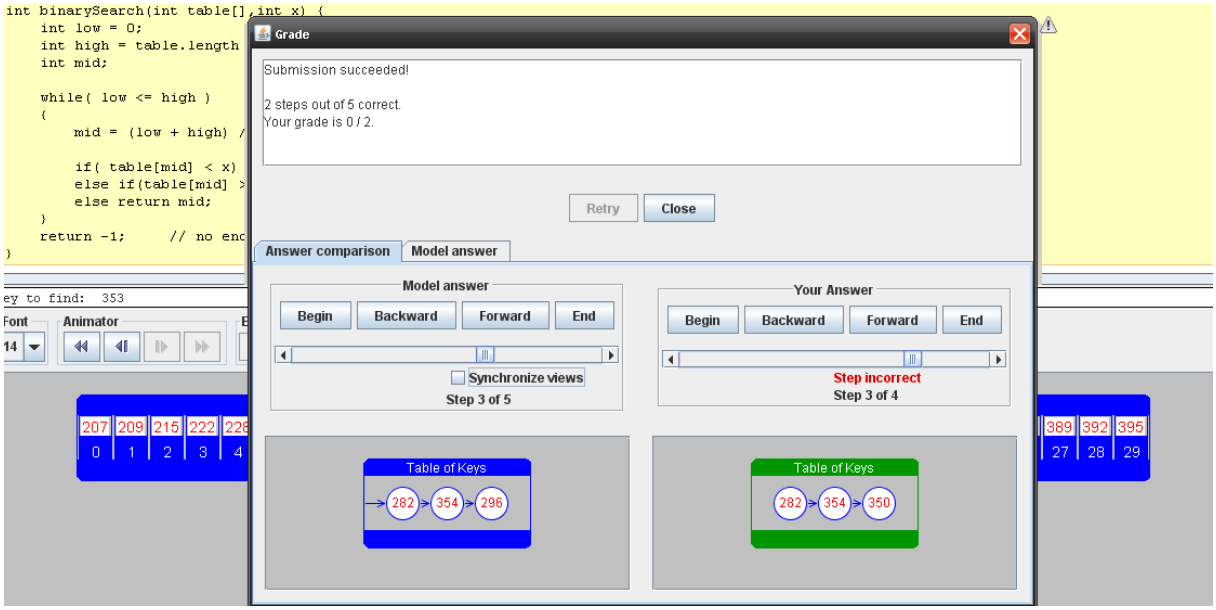


Fonte: Elaborado pelo autor

Depois da parte amarela da figura 18, encontra-se o *label* **animator** para indicar os quatro botões de controle da animação que permitem prosseguir e retroceder passo a passo, ir diretamente para o início da simulação ou para o último passo executado (Requisito II). Cada simulação é acompanhada pelo pseudocódigo e pelas instruções de manuseio dos elementos do applet Java (Requisito III). Depois que o aluno dá por terminada a execução do algoritmo,

deve submeter a solução clicando no botão Submit. Em seguida, pode clicar em Model answer para conferir a solução, como ilustrado na Figura 19:

Figura 19 – Corretor Trakla2



Fonte: Elaborado pelo autor

A figura 19 é dividida em duas partes: a primeira com a solução oficial do sistema e a segunda com a solução submetida pelo aluno. Este recurso, aliado à simplicidade de uso do sistema, significa autonomia para o estudante praticar em casa, já que pode conferir suas respostas e avançar para os próprios exercícios.

Os resultados do trabalho de Kornonen et al. (2009) com TRAKLA2, em um experimento que colocava os estudantes para interagir com as simulações em diferentes níveis, apontaram melhores efeitos na aprendizagem quando o engajamento dos alunos era maior; em outras palavras, quando os alunos tinha liberdade para modificar diretamente as estruturas de dados, ao invés de ficarem como meros expectadores de simulações automáticas, percebeu-se que eles ficavam mais imersos nas tarefas e encaravam os problema de uma forma mais abrangente.

6.5 BUSCA E SELEÇÃO DOS DEPURADORES

Por se tratar de ferramentas para o ensino de programação, os depuradores ou debuggers selecionados devem mostrar os conceitos abstratos da execução dos algoritmos através de representações visuais concretas, bem como garantir requisitos de qualidade que os tornem fáceis de usar, de tal forma que o estudante gaste pouco tempo com dúvidas sobre a operação.A busca, entretanto, encontrou apenas um depurador construído para fins

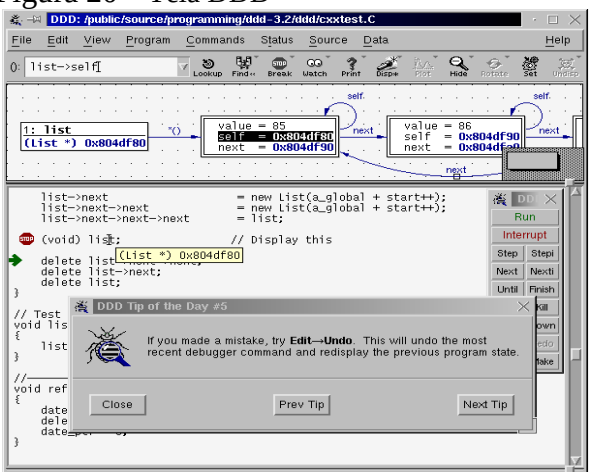
pedagógicos, todos os demais são ferramentas projetadas para uso profissional, com uma complexidade de uso inadequada para iniciantes. Por outro lado, notou-se que alguns dos simuladores apresentados na seção 6.4.2 implementam bem a funcionalidade de depuração; além disso, percebeu-se que o depurador verificado e selecionado para compor o conjunto de artefatos dá suporte para visualizações de leitura e escrita de variáveis e de laços de repetição, que é o necessário para auxiliar na codificação dos primeiros programas. A seleção dessa ferramenta considerou os requisitos de qualidade definidos na seção 6.4.

6.5.1 Alguns dos depuradores verificados e não selecionados

O fato de a maioria dos depuradores verificados ser destinada para uso comercial tais como o Eclipse da Sun e Visual Studio da Microsoft, aponta em direção ao pensamento de que procurar erros ou observar minuciosamente o comportamento indica um nível razoável de maturidade; em outras palavras, programas de depuração mais simples ou os próprios apllets simuladores são mais indicados para mostrar as operações dos algoritmos de forma mais pedagógica, uma vez que os debuggers profissionais, além de não serem voltados para o ensino, são complexos para serem utilizados por iniciantes.

Um dos depuradores encontrados é o DDD – Data Display Debugger – uma interface gráfica para interação com o usuário para depuradores de linha de comando. Ele é licenciado através da GPL. Uma grande vantagem do DDD está no fato de exibir as variáveis utilizadas pelo programa, juntamente com seu conteúdo, além de ponteiros com suas referências e estruturas de dados. Porém, para uso no ensino de linguagem de programação é uma ferramenta que demanda treinamento, pois é necessário compilar o programa para fazer a depuração; outro agravante é o curto espaço de visualização dos elementos gráficos numa área retangular. A Figura 20 ilustra a interface da ferramenta:

Figura 20 – Tela DDD

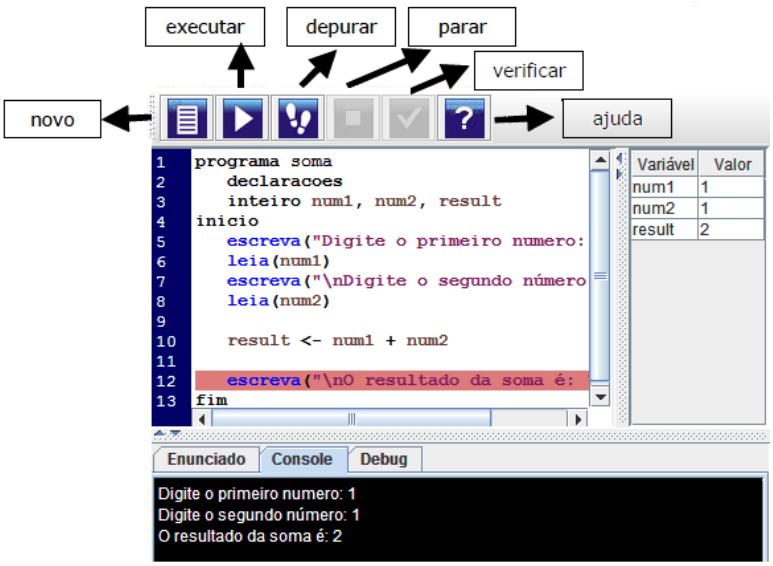


6.5.2 Depurador selecionado: webportugol

WebPortugol, apresentada no trabalho de Hostin e Raabe (2007), é uma ferramenta de auxílio à aprendizagem de lógica de programação com codificação na linguagem portugol. A ferramenta possui uma interface simples e permite apenas a construção de algoritmos com as operações fundamentais, deixando de fora conceitos mais complexos como funções, recursividade e ponteiros. Assim, é indicada para a parte inicial de um curso de programação introdutória.

O destaque de *WebPortugol* é o recurso de depuração passo a passo que permite ao estudante visualizar a escrita de valores em variáveis, possibilitando assim a descoberta de erros e o aprimoramento do entendimento do comportamento dos algoritmos (Requisito III). A Figura 21 ilustra a ferramenta com seus recursos:

Figura 21 – Web Portugol



Fonte: Elaborado pelo autor

Pode- se ver pela figura 8 que o modo de operação da ferramenta é simples de tal forma que o estudante saiba instantaneamente como rodar os algoritmos (Requisito I). Na parte superior, há os botões **novo**, que limpa o console para a escrita de um novo algoritmo, **executar**, que roda o algoritmo, **depurar**, que aciona a execução passo a passo, **parar**, que quebra a execução do programa. Estes botões permitem que o usuário controle a sequência de utilização da ferramenta e que corrija suas respostas (Requisitos III). Ainda na parte de cima da imagem, há o botão ajuda que abre direciona para uma janela que contém todos os comandos e tipos de dados implementados pela ferramenta.

6.6 VINCULAÇÃO DO CONJUNTO DE ARTEFATOS FINAL AO CONTÉUDO DE PROGRAMAÇÃO INTRODUTÓRIA

A seleção do conjunto de artefatos educacionais foi realizada para ser aplicada em todas as etapas da disciplina de Fundamento de Programação do curso de Sistemas de Informação da Universidade Federal do Ceará; procurou-se, portanto, uma solução que permitisse aos professores de algoritmos ter um recurso auxiliar do início ao fim do conteúdo programático, dentro de uma sequência planejada, conforme o modelo apresentado na seção 5.2. Assim, de posse do conjunto final de artefatos educacionais e com a definição de sua sequência de utilização, chegou-se ao Quadro 6 que apresenta cada categoria de artefato educacional aplicada à ementa de programação introdutória considerada:

Quadro 6 – Artefatos educacionais distribuídos na ementa de Fundamentos de Programação

UNIDADES E ASSUNTOS DA AULA	Artefatos Educacionais				
	Jogos	Exercícios Contextualizados	Simuladores	Debuggers	Linguagens visuais
1. Introdução, apresentação do curso e resolução de exercícios de nivelamento Noções de programação: PORTUGOL (atribuições, variáveis, tipos primitivos, boas práticas)	x	x			x
2. Noções de programação e estruturas de seleção.	x	x			
3. Noções de programação e estruturas de repetição; Exercícios	x	x	x	x	x
4. Variáveis Globais, Locais e Estáticas		x		x	x
5. Programação estruturada: Funções	x	x	x	x	x
6. Programação estruturada: Funções. Recursividade. Pilha de execução. Exercícios	x	x	x		
7. Ponteiros.		x	x		
8. Matrizes Unidimensionais; Exercícios		x	x		
9. Matrizes Bidimensionais; Exercícios		x	x		
10. Cadeia de caracteres		x			
11. Alocação dinâmica; Exercícios; Definição do trabalho 3		x	x		
12. Tipos Estruturados; Exercícios		x	x		
13. Tipos Abstratos de Dados; Exercícios		X	x		

Fonte: Ementa do curso de programação Introdutória

Este Quadro significa uma visão dos artefatos educacionais dentro do conteúdo programático de um curso de fundamentos de programação. A partir dessa visão, cada professor pode planejar à sua maneira a melhor forma de inserir ferramentas visuais e exercícios contextualizados em suas aulas, além de poder mensurar empiricamente o valor que os artefatos estão agregando ao processo de ensino e aprendizagem em cada etapa do conteúdo. Para dar um ponto de partida aos docentes, a pesquisa não apenas definiu e seqüenciou o que usar, mas também propôs quais artefatos utilizar; em outras palavras, o

trabalho reuniu um conjunto de ferramentas prontas para uso, de acordo com os critérios de qualidade definidos nas seções 2.5.1 e 2.5.2. Assim, chegou-se ao Quadro 7.

Quadro 7 – Conjunto de artefatos educacionais selecionados

Artefatos educacionais	Identificador	Aplicabilidade	Fonte
Jogos	LIGHT BOT 2	Comandos de seleção, repetição, recursividade, funções, dividir para conquistar, otimização, modularização	http://armorgames.com/play/6061/light-bot-20
Exercícios contextualizados	LISTA DE EXERCÍCIOS SPOJ	Laços de repetição, comandos de decisão, matrizes, funções	br.spoj.com
Depuradores	WEB PORTUGOL	Variáveis, comandos de decisão, laços de repetição, operadores lógicos	http://siaiacad17.univali.br/webportugol/
Ling. Pedagógicas	SCRATCH	Variáveis, laços de repetição, funções	http://scratch.mit.edu/
Simuladores	SORT	Algoritmos de ordenação: Buble, Selection, Insertion, Merge, matrizes	http://www.csse.monash.edu.au/~dwa/Animations/
	SIMULADOR PONTEIROS EM C	Aritmética de ponteiros; funções malloc(), realloc(); free().	http://www.pitt.edu/~is2470pb/Fall02/Final/rm/interactive.html
	TRAKLA2	Algoritmos básicos: busca binária, busca por interpolação, avaliação de expressões posfixa e infixa com pilha, programação dinâmica, algoritmos de ordenação. Algoritmos para estruturas de dados, recursividade	https://trakla.cs.hut.fi/app

Fonte: elaborado pelo autor.

O Quadro apresenta a seleção do conjunto de artefatos proposto por este trabalho para um curso de Programação Introdutória, juntamente com a informação dos conceitos que cada artefato educacional aborda.

7 CONSIDERAÇÕES FINAIS

Destacou-se na introdução da pesquisa a realidade das altas taxas de reprovação nas disciplinas de programação e algoritmos, o que contribui para elevar a evasão nos cursos de T.I e Ciências da Computação. Ao investigar o problema, o estudo encontrou na revisão da literatura um discurso convergente sobre as grandes dificuldades dos alunos para entender abstrações e resolver problemas, juntamente com propostas de uso de ferramentas visuais para melhorar tanto o nível de aprendizagem como o desempenho nas provas.

A despeito de fatores importantes, tais como a aplicabilidade da computação na vida real e a motivação, percebeu-se que há ainda uma grande tendência em tomar as notas

das avaliações dos alunos como única referência da aprendizagem. É certo que a nota é um grande indicador do que o aluno aprendeu nas aulas, porém essa avaliação mostra apenas o perfil técnico dele, sem levar em conta a postura e a compreensão do mesmo em relação ao que a computação representa na realidade. Levando em conta este cenário pedagógico, o conjunto de artefatos educacionais selecionado por este estudo foi concebido não só para ajudar a elevar o desempenho dos estudantes nas provas, mas também para ajudá-los a compreender a nobreza da computação e, assim, mantê-los motivados para enfrentar os desafios até a conclusão do curso.

Outra constatação importante da pesquisa é que a maioria das propostas sobre o uso de jogos e ferramentas se restringe à parte inicial do curso, abordando apenas os primeiros conceitos fundamentais e o segundo escopo de problema (dificuldades no entendimento de abstrações) definido no trabalho. Esta abordagem é justificável, pois a forma como se inicia a disciplina de programação pesa bastante no desempenho dos estudantes; por outro lado, é muito proveitoso também utilizar recursos de softwares na fase intermediária das disciplinas para auxiliar e dinamizar as aulas. Foi justamente pensando em garantir o uso de ferramentas de software, num maior nível de abrangência, que se definiu o intuito do estudo: combinar as melhores soluções de informática na educação para o ensino de algoritmos.

Uma das dificuldades que tornou o estudo delicado foi a consideração do fator aprendizagem, que é algo difícil de mensurar. Segundo Savi (2011, p.194), "avaliar a aprendizagem é um procedimento bastante complexo". O mesmo autor destaca que é essencial utilizar a estratégia de autoavaliação do aluno, ou seja, detectar os efeitos dos jogos e ferramentas a partir da percepção dele. Cientes dessa complexidade, os autores pesquisados procuraram captar o *feedback* dos próprios alunos para avaliar a influência dos artefatos educacionais sobre a aprendizagem. Assim, o trabalhou levou em conta os casos de sucesso ou pelo menos os aspectos positivos de cada experiência, bem como os requisitos de qualidade para ferramentas educacionais, como referência para a seleção do conjunto de artefatos educacionais. Mesmo assim, uma avaliação prática de cada artefato selecionado ou pelo menos de uma parte do conjunto teria agregado mais autoridade à seleção.

Com relação à busca dos artefatos educacionais, definiu-se de forma muito clara o que se procurava, de acordo com critérios pedagógicos e técnicos; porém, um número pequeno de opções para a escolha de jogos atrativos e adequados ao público, bem como a dificuldade para encontrar ferramentas visuais para vários conceitos, tornaram limitada a seleção do conjunto de artefatos final. Grande parte dos jogos gratuitos encontrados tinha um enfoque para o público infantil; alguns jogos melhores eram pagos. Com relação aos

simuladores encontrados na web, a maioria é para algoritmos de ordenação. Porém, para compensar o esforço da busca, o trabalho encontrou o ambiente TRALAK2, que contém mais de 50 exercícios de simulação, envolvendo algoritmos básicos e estruturas de dados.

7.1 RESULTADOS E TRABALHOS FUTUROS

Procurou-se entender as razões por trás das dificuldades dos alunos com programação através de uma profunda revisão da literatura, que culminou na categorização dos problemas identificados em dois escopos de problemas: raciocínio lógico para a interpretação e resolução de problemas e dificuldades com abstrações e visualização das operações dos algoritmos. Em outras palavras, não só se chegou à identificação dos problemas, com também se realizou a classificação dos mesmos já na introdução do trabalho.

A partir das várias publicações em Simpósios e Workshops de Informática na Educação, a pesquisa pôde selecionar as principais propostas criadas para auxiliar os alunos na aprendizagem de algoritmos e lógica de programação. Notou-se que a maioria dessas propostas fazia uso isolado de apenas um recurso de software ou considerava apenas a parte inicial da disciplina de programação introdutória. Assim, para preencher essa lacuna, o trabalho conseguiu propor um modelo para a combinação dos artefatos educacionais. Esse modelo, definido na **seção 5.2**, resultou na criação do Quadro 6 **seção 6.6**, que mostra a distribuição dos tipos de artefatos educacionais aplicados à ementa de programação introdutória.

Ao tomar conhecimento sobre os principais problemas e soluções no processo de ensino e aprendizagem de programação, a pesquisa realizou com muita clareza a busca e a seleção dos artefatos educacionais, o que resultou no conjunto de artefatos educacionais reunidos no Quadro 7 da **seção 6.6**. Chegou-se, portanto, a uma solução que pretende combinar e utilizar os pontos fortes das melhores soluções de informática na educação para o ensino de algoritmos.

Vale ser lembrando aqui que o estudo se fundamentou na premissa de que é fundamental iniciar as disciplinas de programação, proporcionando um ambiente positivo e motivador, de tal forma que os estudantes ganhem energia suficiente para prosseguirem até o fim. Além disso, chegou-se à conclusão de que é importante planejar um ensino mais gradual que diminua o impacto da mudança de paradigma de pensamento. Por ultimo, a revisão da literatura apontou para o fato de que é essencial levar em conta não apenas os aspectos técnicos, mas também os sentimentos e a postura dos estudantes em relação à computação.

Trabalhos futuros podem criar novas combinações de artefatos educacionais que levem em conta os dois escopos de problemas definidos no trabalho, de tal forma que fique evidenciada a natureza complementar dos artefatos entre si; em outras palavras, uma pesquisa nessa direção, deve evidenciar como o conjunto final quebra a problemática do ensino e aprendizagem de programação em problemas menores, distribuindo os artefatos educacionais nas etapas de um curso de programação introdutória. Assim um trabalho nesse sentido, pode levar em conta dois objetivos: o primeiro é ampliar a validação da forma como a presente pesquisa enxergou um curso de fundamentos de programação, definindo dois escopos de problemas; e o segundo é trazer um novo conjunto de artefatos educacionais, seguindo uma abordagem igual ou similar à adotada nesta pesquisa.

Outra alternativa é avaliar o conjunto de artefatos educacionais final da pesquisa aplicado a um curso de programação introdutória, com o objetivo de aperfeiçoar tanto a seleção desse conjunto como a seqüência de utilização do mesmo, averiguando-se o impacto na aprendizagem. Na verdade, não há a obrigação de se avaliar exatamente o conjunto ao qual o estudo chegou, assim, outro conjunto pode ser empregado desde que ele obedeça às categorias definidas pelo estudo.

REFERÊNCIAS

AURELIANO, O. C. V.; TEDESCO, R. A. C. P. Ensino-aprendizagem para Iniciantes: Uma revisão sistemática da literatura focada no SBIE e WIE. In: **23º Simpósio Brasileiro de Informática na Educação**, Rio de Janeiro, (2012).

BATTISTELLA, E. P.; WANGENHEIM, V. A.; WANGENHEIM, V. G. C. SORTIA: Um Jogo para o Ensino de Algoritmo de Ordenação na Disciplina de Estrutura de Dados: Estudo de Caso na Disciplina de Estruturas de Dados. In: **23º Simpósio Brasileiro de Informática na Educação**. Rio de Janeiro. 2012.

BARBOSA, S. D. J; SILVA, B. S. Interação Humano-Computador. Elsevier. Rio de Janeiro. 2010.

BORGES, F. A. M. Avaliação de uma metodologia alternativa para a aprendizagem de programação. In: **VIII Workshop de Educação em Computação**, Curitiba. 2000.

BRITO JUNIOR, R. O.. SAED: Simulador de algoritmos e estruturas de dados. Monografia. Universidade Estadual de Montes Claros, Montes Claros. 2006.

COUTINHO, F. R. S.; ALMEIDA, J.; PRATES, R.. O; CHAIMOWICZ, L.. Belesminha: Um jogo educacional para apoio ao aprendizado de recursividade**.** In: **SBGAMES – VII Simpósio Brasileiro de Jogos para Computador e Entretenimento Digital**, Belo Horizonte. 2008.

DANTE. L. R.. Didática da Resolução de Problemas de Matemática. São Paulo. Ática. 2002.

HOSTINS, H.; RAABE, A. Auxiliando a Aprendizagem de Algoritmos com a Ferramenta Webportugol. In: **XV Workshop sobre Educação em Computação.** Rio de Janeiro. 2007.

HU, M. Using Game Scenarios for Teaching Novice Programmers. **Institutes of Technology and Polytechnics (ITP)**: in New Zeland. Journal of Applied Computing & Information Technology. 2008.

JESUS, E. A.; RAABE, A. L. A. Avaliação Empírica da Utilização de um Jogo para Auxiliar a Aprendizagem de Programação. In**: 21º Simpósio Brasileiro de Informática na Educação**, João Pessoa. 2010.

JESUS, E. A.; RAABE, A. L. A. Interpretações da Taxonomia de Bloom no Contexto da Programação Introdutória. In: **20º Simpósio Brasileiro de Informática na Educação**, Florianópolis. 2009.

JÚNIOR, J. C. R. P.; RAPKIEWICZ, C. E. O Processo de Ensino e Aprendizagem de Algoritmos e Programação: Uma Visão Crítica da Pesquisa no Brasil. **III Workshop de Educação em Computação e Informática do estado de Minas Gerais** (WEIMIG). Belo Horizonte. 2004. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/weirjes/2004/003.pdf>. Acesso em: 29 dez. 2012.

KELLEHER, C.; PAUSCH, R. Lowering the Barriers to Programming: survey of programming environments and languages for novice programmers. **Carnegie Mellon University**: in EUA. 2008. Disponível em: <https://www.cs.cmu.edu/~caitlin/papers/NoviceProgSurvey.pdf >. Acesso em: 21 nov. 2014

MARQUES, D. L.; COSTA, L. F. S.; SILVA, M. A. A.; REBOUÇAS, A. D. D. S. Atraindo Alunos do Ensino Médio para a Computação: Uma Experiência Prática de Introdução a Programação utilizando Jogos e Python. In: **17º Workshop de Informática na Escola**, Aracaju. 2011.

MENDES, J. P.; SCAICO, P. D.; LIMA, A. A.; DA SILVA, J. B. B.; AZEVEDO, S; PAIVA, L. F; RAPOSO, E. H. S; ALENCAR, Y. Programação no Ensino Médio: Uma abordagem de Ensino Orientado ao *Design* com Scratch. In: **18º Workshop de Informática na Escola**, Rio de Janeiro. 2012.

MELO, F. E. N; CUNHA, R. R. M; SCOLARO, D. R; CAMPOS, J. L. Do sratch ao arduino: uma proposta para o ensino introdutório de programação para cursos superiores de tecnologia. In: XXXIX Congresso Brasileiro de Educação em Engenharia. Blumenau. 2011.

MORATORI, P. B.. Por que utilizar Jogos Educativos no Processo de Ensino Aprendizagem. Monografia. Universidade Federal do Rio de Janeiro, Rio de Janeiro. 2003. Disponível em: <http://ucbweb2.castelobranco.br/webcaf/arquivos/23678/15577/t_2003_patrick_barbosa_moratori.pdf> Acesso em: 23 set 2014.

MOTA, M. P., BRITO, S. R., MOREIRA, M. P e FAVERO, E. L. Ambiente Integrado à Plataforma Moodle para Apoio ao Desenvolvimento das Habilidades Iniciais de Programação. In: Anais do XX Simpósio Brasileiro de Informática na Educação. 2009.

NETO, W.C. B, SCHUVARTZ, A. A. Ferramenta Computacional de Apoio ao Processo de Ensino-Aprendizagem dos Fundamentos de Programação de Computadores. In**: XVIII Simpósio Brasileiro de Informática na Educação**, Mackenzie. 2007.

PELIZZARI, A. Pedagogia de projetos: uma forma de garantir a aprendizagem significativa. Tese. Universidade Federal de Santa Catarina, Florianópolis. 2003. Disponível em: <https://repositorio.ufsc.br/handle/123456789/86537> Acesso em: 15 set 2014.

PEREIRA, J. C. R., RAPKIEWICZ, C. E., DELGADO, C., XEXEO, J. A. M. Ensino de Algoritmos e Programação: Uma Experiência no Nível Médio. **XIII Workshop de Educação em Computação**. São Leopoldo, RS. 2005.

PÍCCOLO, H. L.; SENA, V. de F; NOGUEIRA, K. B.; SILVA, M. O.; MAIA, Y. A. N. Ambiente Interativo e Adaptável para ensino de Programação. **In: 21º Simpósio Brasileiro de Informática na Educação**, João Pessoa. 2010.

PIMENTEL, E. P.; FRANÇA, V. F. de; NORONHA, R. V.; OMAR, N. Avaliação Contínua da Aprendizagem, das Competências e Habilidades em Programação de Computadores. In: **9º Workshop de Informática na Escola**, Campinas. 2003.

PIVA JUNIOR, D.; FREITAS, R. L. Estratégias para melhorar os processos de abstração na disciplina de algoritmos. In**: 21º Simpósio Brasileiro de Informática na Educação**, João Pessoa. 2010.

POLYA, G.. A Arte de Resolver Problemas. Rio de Janeiro. Interciência. 1975.

RAPKIEWICZ, C. E; FALKEMBACK, G; SEIXAS, L; dos Santos, N. S, Cunha, V. V, Klemann, M. Estratégias pedagógicas no ensino de algoritmos e programação associadas ao uso de jogos educacionais. **Ciclo de Palestras Novas Tecnologias na Educação** - UFRGS, Porto Alegre, 2006.

PIVA JUNIOR, D.; FREITAS, R. L. Estratégias para melhorar os processos de abstração na disciplina de algoritmos. In**: 21º Simpósio Brasileiro de Informática na Educação**, João Pessoa. 2010.

RIBEIRO, R.S; BRANDÃO, L. O; BRANDÃO, A. A. F. Uma visão do cenário Nacional do Ensino de Algoritmos e Programação: uma proposta baseada no Paradigma de Programação Visual. In**: 23º Simpósio Brasileiro de Informática na Educação**, Rio de Janeiro. 2012.

SALES, C. G, DANTAS, V. F. ProGame: um jogo para o ensino de algoritmos e programação. In**: 21º Simpósio Brasileiro de Informática na Educação**, João Pessoa. 2010.

SANTOS NETO, V., M. A utilização da ferramenta scracth como auxílio na aprendizagem de programação. **II Congresso Brasileiro de Informática na Educação (CBIE**). 2013**.**

SANTOS, R. P. e COSTA, H. A. X. TBC-AED e TBC-AED/WEB*:* Um Desafio no Ensino de Algoritmos, Estruturas de Dados e Programação. In: **IV Workshop de Educação em Computação e Informática do estado de Minas Gerais** (WEIMIG). Varginha, MG. 2005.

SANTOS, R. P.; COSTA, H. A. X. Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos Iniciantes em Computação e Informática. INFOCOMP (UFLA), Lavras - MG, v. 5, n. 1, p. 41-50, 2006.

SANTOS, R. M. S.; COSTA, F. A. A.. Ensino da Programação Através de Programação Visual. Tese. Universidade de Lisboa, Lisboa, Portugal. 2013. Disponível em: < http://repositorio.ul.pt/handle/10451/9137 > Acesso em: 23 set 2014.

SAVI, R. Avaliação de jogos voltados para a disseminação do conhecimento. Tese. Universidade Federal de Santa Catarina, Florianópolis. 2011. Disponível em: < http://btd.egc.ufsc.br/wp-content/uploads/2011/12/RafaelSavi.pdf>. Acesso em: 20 dez 2012.

SOARES, T. C. A. P.; CORDEIRO E. S.; STEFANI Í. G. A.; TIRELO, F. Uma Proposta Metodológica para o Aprendizado de Algoritmos em Grafos Via Animação Não-Intrusiva de Algoritmos. In: **III Workshop de Educação em Computação e Informática do Estado de Minas Gerais** (WEIMIG' 2004). Belo Horizonte, MG, Brasil. Disponível em: <http://www.italost.com/academic/papers/pdf/ring_weimig2004.pdf>. Acesso em: 16 jan 2013.

SOUZA, F. R.; GIRALDI, M. C. M. Giraffas - Uma ferramenta de apoio ao ensino da estrutura de dados árvore. In: **22º Simpósio Brasileiro de Informática na Educação**, Aracajú, 2011.

TAROUCO, L. M. R.; ROLAND, L. C.; FABRE, M. C. J. M. KONRATH, M. L. P. Jogos educacionais. CINTED/UFRGS. 2004. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/weirjes/2004/003.pdf >. Acesso em 20 dez 2012.

TAVARES, R. Ambiente colaborativo on-line e a aprendizagem significativa de Física. **13º Congresso Internacional de Educação a Distância – ABED**. Curitiba – PR, 2007. Disponível em <http://www.fisica.ufpb.br/~romero/pdf/2007ABEDRomero.pdf> Acesso em: 15 set 2014.

TAVARES, W. Classificação de Problemas do SPOJ. [notas de aula]. Quixadá: UFC, 2012.

VALASKI, J.; PARAISO, C. E. Limitações da Utilização do Alice no Ensino de Programação para Alunos de Graduação. In: **23º Simpósio Brasileiro de Informática na Educação**, Rio de Janeiro. 2012.

VALENTIM, H. Um estudo sobre o ensino-aprendizagem de lógica de programação. **In: VII Encontro Nacional de Pesquisas em Educação em Ciências, Florianópolis**. 2009.

VILAÇA, M. L. C. Pesquisa e ensino: considerações e reflexões. UNIGRANRIO. Revista do Curso de Letras da UNIABEU, Nilópolis, v. I, Número 2, Mai. Ago. 2010. Disponível em: <

http://www.unisc.br/portal/upload/com_arquivo/pesquisa_e_ensino_consideracoes_e_reflexoes.pdf> Acesso em 15 out 2014.

.

APÊNDICES

APÊNDICE A – Classificação dos Problemas SPOJ – uma síntese

Leitura e Saída

http://br.spoj.com/problems/QUADRAD2/
http://br.spoj.com/problems/JPNEU/
http://br.spoj.com/problems/TRANSP11/

Comando de seleção

http://br.spoj.com/problems/FROTATAX/
http://br.spoj.com/problems/JSEDEX/
http://br.spoj.com/problems/CONTA1/
http://br.spoj.com/problems/TRIANG11/
http://br.spoj.com/problems/NOTA09/
http://br.spoj.com/problems/PAPEL09/
http://br.spoj.com/problems/OVERF09/

Comando de repetição

http://br.spoj.com/problems/SOMA/
http://br.spoj.com/problems/POODLEMG/
http://projecteuler.net/problem=1
http://br.spoj.com/problems/MARAT09/
http://br.spoj.com/problems/FUGIT09/
http://br.spoj.com/problems/JTACOGRA/
http://br.spoj.com/problems/COMETA2/

http://br.spoj.com/problems/JGARCOM/
http://br.spoj.com/problems/METEORO/
http://br.spoj.com/problems/QUERM/
http://br.spoj.com/problems/AERO/ (Encontrar o número que mais se repete)
http://br.spoj.com/problems/COFRE/
http://br.spoj.com/problems/ESTAGIO/
http://br.spoj.com/problems/METEORO/
http://br.spoj.com/problems/PAR/
http://br.spoj.com/problems/BAFO/
http://br.spoj.com/problems/FATORIA2/
http://br.spoj.com/problems/PRIMO/

Operador módulo

http://br.spoj.com/problems/BIT/
Matriz
http://br.spoj.com/problems/MINHOCA/
http://br.spoj.com/problems/MAGICO11/
http://br.spoj.com/problems/MITO09/
Vetor
http://br.spoj.com/problems/TRILHAS/
http://br.spoj.com/problems/PARPROX/
http://br.spoj.com/problems/MINADO12/

Recursão

http://br.spoj.com/problems/ACOES1MG/
http://br.spoj.com/problems/F91/
String
http://br.spoj.com/problems/ENCOTEL/
http://br.spoj.com/problems/PAPRIMAS/
http://br.spoj.com/problems/ONZE/
http://rosalind.info/problems/dna/
http://rosalind.info/problems/rna/
http://rosalind.info/problems/revc/