AAPW: Uma ferramenta para facilitar o aprendizado de programação Web

Mouglas E. N. Gomes[1], Richarlyson A. D'Emery[2], Gilberto A. A. C. Filho[1]

[1]Departamento de Estatística e Informática (DEINFO) – Universidade Federal Rural de Pernambuco (UFRPE), Caixa Postal 560 – 52171-900 – Recife – PE – Brasil

[2]Unidade Acadêmica de Serra Talhada (UAST) – Universidade Federal Rural de Pernambuco (UFRPE) – Serra Talhada – PE – Brasil

{mowgllasx8, g.cysneiros}@gmail.com, rico_demery@yahoo.com.br

Abstract. *Based on the teaching-learning process in websites development and of the faced difficulties in this process, this paper presents the AAPW (Learning Environment for Web Programming, in Portuguese called Ambiente de Aprendizagem de Programação Web), intended for learning programming for Web development and functional code generation for websites. It is also presented a discussion on the compiler's phases as well as APPW characteristics. An experimental research with an exploratory and descriptive approach was conducted by means of a case study in order of validating the first version of AAPW.*

Resumo. *Tomando por base o processo de ensino-aprendizagem no desenvolvimento de websites e das dificuldades enfrentadas nesse processo, este trabalho apresenta a ferramenta AAPW (Ambiente de Aprendizagem de Programação Web), que possibilita o aprendizado de programação para desenvolvimento Web e a geração de codificação funcional de websites. Também é apresentada uma discussão sobre as fases envolvidas em compiladores e sobre as características da ferramenta. Foi realizada uma pesquisa de abordagem experimental e caráter exploratório-descritivo, com o emprego de um estudo de caso, visando validar a primeira versão da AAPW.*

1. Introdução

Com a expansão da Internet surgiram várias tecnologias para a otimização e criação de *websites*, e muitos são os que buscam tal mercado, porém encontram muitas dificuldades, em virtude da complexidade das soluções para o desenvolvimento Web.

São diversas as linguagens de programação voltadas ao desenvolvimento de *websites*, e algumas são de difícil compreensão. Apesar da existência de ferramentas para tal propósito como, por exemplo, Dreamweaver[1], Eclipse[2], Visual Studio[3], entre outras, muitas são proprietárias e mesmo com todas as facilidades, a codificação gerada é confusa, o que dificulta a compreensão aos iniciantes em programação.

[1] Disponível em <http://www.adobe.com/br/products/dreamweaver.html>
[2] Disponível em <http://www.eclipse.org/>
[3] Disponível em <http://www.microsoft.com/visualstudio/11/pt-br/products/visualstudio>

Para Sebesta (2006), uma linguagem com um grande número de componentes "é bem mais difícil de ser entendida que outra com poucos componentes". São escassas as ferramentas que auxiliam o processo ensino-aprendizagem envolvendo programação Web. Neste contexto, este trabalho propõe a criação da ferramenta AAPW (Ambiente de Aprendizagem de Programação Web).

AAPW permite o aprendizado de conceitos envolvidos na programação Web, de forma simples e objetiva, podendo ser utilizada como recurso auxiliar em disciplinas de cursos que abordam desenvolvimento Web. Ela prevê o entendimento nos lados cliente e servidor, tratamento de formulários, elimina a necessidades da inclusão de *tags* no código e utiliza instruções simples com comandos em português.

2. Referencial Conceitual

2.1. Conceitos Básicos

Segundo Menezes (2005), um programa é um **procedimento efetivo** descrito em qualquer formalismo o qual descreva procedimentos possíveis a serem executados por um computador. Menezes ainda afirma que um procedimento efetivo pode ser definido como "um programa que possui uma sequência finita de instruções e que podem ser executadas mecanicamente em tempo finito".

Uma linguagem é um conjunto de palavras sobre um alfabeto, tem uma sintaxe bem definida que permite uma sentença ser identificada como pertence ou não a determinada linguagem. A semântica deve ser precisa, como nas linguagens Java[4], C[5] e HTML[6] (*HyperText Markup Language*), por exemplo [Price e Toscani 2005].

Uma linguagem é representada por **gramáticas**, as quais consistem em um dispositivo que a partir de uma sentença é gerada uma linguagem. Para Sebesta (2006), a gramática BNF (*Backus–Naur Form*) é uma notação natural para descrever sintaxes, sendo o método mais popular para descrever concisamente a sintaxe de uma linguagem de programação; é considerada uma metalinguagem, ou seja, uma linguagem utilizada para descrever outras linguagens.

Expressão regular é uma maneira de descrever conjuntos regulares. É utilizada em construção de compiladores, editores, sistemas operacionais, protocolos, etc. Trata-se de um formalismo denotacional, também considerado gerador, que permite a inferência sobre a construção ou geração de palavras pertencentes a uma linguagem. Uma expressão regular é definida a partir de conjuntos básicos e operações de concatenação e união. Toda linguagem regular pode ser descrita por uma expressão regular e é definida a partir de conjuntos de linguagens básicas e que possuem uma álgebra bem definida [Menezes 2005].

2.2. Compiladores

Segundo Aho et al. (1995) o propósito básico das linguagens de programação é gerar instruções para que o computador possa executar o processamento. Existe uma grande

[4] Disponível em <http://www.oracle.com/br/technologies/java/index.html>
[5] Disponível em <http://www.cprogramming.com>
[6] Disponível em <http://www.w3.org/html/>

necessidade em facilitar a comunicação entre seres humanos e computadores, visto que os computadores operam em linguagem de máquina, mais especificamente em base binária, o que é extremamente complexo e sujeito a erros por seres humanos. É preferível, nesse caso, uma linguagem mais próxima a comunicação humana. A Figura 1 ilustra a estrutura geral dos compiladores.

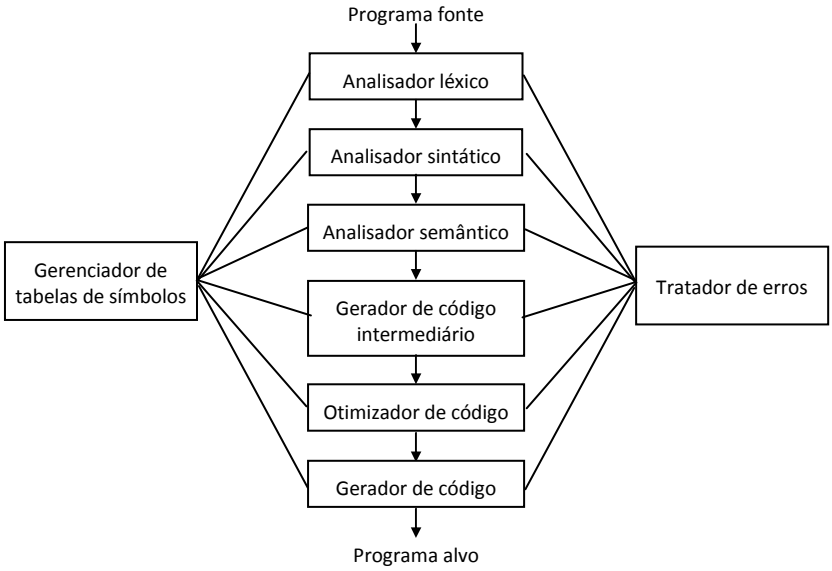

Figura 1. Estrutura de um compilador

Na Figura 1 observa-se que um programa fonte é submetido a várias camadas até a geração do programa alvo. A seguir são apresentadas as principais fases dos compiladores, as quais foram utilizadas para o desenvolvimento da ferramenta proposta neste trabalho.

O **analisador léxico** é a primeira fase que compõe o processo de compilação de um texto fonte (*source*). Executa uma série de tarefas, sendo a principal delas a de fragmentar o programa fonte de entrada em trechos elementares completos com identificação própria, ou seja, com um tipo de ID [Price e Toscani 2005].

A **análise sintática** é o segundo componente dos compiladores. Tem a função de promover a análise da sequência com que os elementos do código fonte se apresentam, a partir da qual se efetua a síntese da árvore sintática, com base na linguagem fonte. A análise sintática evidencia exclusivamente a forma das sentenças da linguagem, e a partir de uma gramática, se fornece a estrutura sintática [Aho et al. 1995].

A **análise semântica** realiza a interpretação do texto fonte. Compõe a terceira etapa do processo de compilação e se refere à tradução do programa fonte em programa objeto. Seu objetivo é capturar o significado das ações a serem executadas a partir do texto fonte. A fase da análise semântica é de difícil formalização, exigindo notações complexas, por isso a maioria das linguagens de programação adota especificações mais simplificadas, com bases informais, através de linguagens naturais [Aho et al. 1995].

Segundo Aho et al. (1995), a entrada para uma rotina de **geração de código** é um programa em linguagem intermediária e a saída do gerador de código é o programa-objeto. Esta saída é tida em distintas maneiras: linguagem de máquina absoluta, linguagem de máquina relocável, linguagem montadora (*assembly*) ou em qualquer

linguagem de programação. A produção de uma linguagem de alto nível simplifica ainda mais a geração de código. Este tipo de implementação atua como pré-processador, e transfere os problemas de geração do código de máquina para outro compilador.

2.2.1. Ferramentas

Existem algumas ferramentas para realizar as tarefas das análises léxica e sintática, como, por exemplo, LEX[7], YACC[8] e GALS[8] (Gerador de Analisadores Léxicos e Sintáticos).

O LEX é um gerador de analisadores léxicos. Trata-se originalmente de um programa para o sistema operacional Unix que lê uma especificação léxica para linguagem de programação C, entretanto, são disponibilizadas versões para outros sistemas operacionais e analisadores para outras linguagens [Lesk 1975].

YACC é um gerador de analisadores sintáticos que pode ser executado em Unix. O analisador gerado é compatível às rotinas de análise léxica gerada por LEX, permitindo a integração do analisador sintático com o léxico gerado por LEX. Dá suporte apenas a Unix e a linguagem de programação C.

A maioria dos geradores disponíveis limita-se, apenas, ao analisador léxico ou ao sintático, podendo ocasionar imprecisões na integração da codificação gerada, uma vez que se faz necessário a utilização de ferramentas distintas. Entretanto, GALS é uma ferramenta para geração de analisadores tanto léxicos quanto sintáticos.

GALS é desenvolvido em Java e pode ser utilizado em ambientes que possuam uma máquina virtual Java. A geração dos analisadores léxicos e sintáticos é realizada através de especificações léxicas baseadas em expressões regulares e especificações sintáticas baseadas em gramáticas livres de contexto, respectivamente [Gesser 2003].

3. Trabalhos Relacionados

A utilização de tecnologias no processo ensino-aprendizagem ganha cada vez mais destaque. Em áreas envolvendo programação, são observadas diversas iniciativas, como, por exemplo, VisualG[9], Jeliot[10] e AMBAP[11].

VisualG é um programa que edita e interpreta algoritmos utilizando uma linguagem próxima do português estruturado, enquanto que na interface gráfica do Jeliot é disponibilizada uma animação ilustrando a interação entre os comandos do código fonte em Java. O AMBAP (Ambiente de Aprendizado de Programação) é um ambiente de aprendizado de programação que visa facilitar o ensino de lógica de programação, mais especificamente, à construção de algoritmos [Almeida et. al 2004].

Apesar da existência de várias ferramentas, não foram observadas ferramentas com o foco no ensino de programação Web, envolvendo o tratamento de *tags* HTML, gerenciamento de servidores Web, entre outras características.

[7] Disponível em <http://www.combo.org/lex_yacc_page/>
[8] Disponível em <http://gals.sourceforge.net>
[9] Disponível em <http://www.baixaki.com.br/download/visualg.htm>
[10] Disponível em <http://cs.joensuu.fi/jeliot/downloads.php>
[11] Disponível em <http://www.ufal.br/tci/ambap>

4. Arquitetura e funcionalidades da ferramenta

Para o desenvolvimento da ferramenta AAPW, primeiramente foram propostas duas gramáticas seguindo o modelo BNF, sendo uma destinada ao compilador (que trata a codificação do lado cliente) e outra para tratar a codificação do lado servidor das aplicações. As gramáticas são ilustradas nas Figuras 2 e 3, respectivamente.

```
//declara o corpo do programa
<programa>::= OP_inicio OP_destino palavras  <corpo>  OP_fim | î;
<corpo>::=<exibir> <texto> <maiscorpo> | <exibir> <area> <maiscorpo> |
<exibir> <radio> <maiscorpo> | <exibir> <selecao> <maiscorpo> | î;
<maiscorpo>::= <corpo>;
//caixa de Texto
<texto>::=OP_texto palavras OP_virgula;
//Area de Texto
<area>::=OP_area palavras OP_virgula;
//botão de radio
<radio>::=OP_radio palavras OP_nome palavras OP_virgula;
//lista de seleção
<opcao>::=OP_opcao palavras OP_valor palavras <maisopcao> | î;
<maisopcao>::=<opcao>;
<selecao>::=OP_selecione OP_nome palavras <opcao> OP_selecione;
//exibir frases na pagina
<exibir>::= OP_exibir OP_paEsquerdo palavras OP_paDireito OP_virgula <maisexibir> |
 OP_exibir OP_paEsquerdo OP_aspa palavras OP_aspa OP_paDireito OP_virgula <maisexibir> | î;
<maisexibir>::= <exibir>;
```

Figura 2. Gramática do lado cliente

```
//declara o corpo do programa
<programa>::= <variaveis> <comandos> | <estrutura_corp> | î;
// declara as variaveis
<variaveis>::= <numeros> <maisvariaveis> | <palavras> <maisvariaveis> | <obter> <maisvariaveis>;
<maisvariaveis>::= <variaveis>  | î;
<numeros>::= OP_numero palavras OP_virgula | OP_numero palavras OP_igual palavras OP_virgula;
<palavras>::= OP_palavra palavras OP_virgula | OP_palavra palavras OP_igual OP_aspa palavras OP_aspa OP_virgula;
<obter>::= OP_palavra palavras OP_igual OP_obter OP_paEsquerdo OP_aspa palavras OP_aspa OP_paDireito OP_virgula;
//estruturas de controle
<comandos>::= <se> | <enquanto> | <exibir> | <expressão>;
<se>::= OP_se OP_paEsquerdo palavras <OP_logico> palavras  OP_paDireito OP_chaEsquerda <estrutura_corp> OP_chaDireita <maisse>;
<maisse>::=<se> | <enquanto> | î;
<enquanto>::=OP_enquanto OP_paEsquerdo palavras <OP_logico> palavras OP_paDireito OP_chaEsquerda <estrutura_corp> OP_chaDireita
<maisenquanto>;
<maisenquanto>::=<enquanto> | <se> | î;
//declaração das logicas
<OP_logico>::= OP_maior | OP_menor | OP_igualigual | OP_maiorigual | OP_menorigual;
<estrutura_corp>::=<exibir> | <expressão> | <expressão> <exibir> | <se> | <enquanto> | <expressão> <exibir> <se> |
<expressão> <exibir> <enquanto>;
//declaração de expressões
<expressão>::= palavras OP_igual palavras <OP_matematico> palavras OP_virgula <maisexpressão>;
<maisexpressão>::= <expressão> | î;
//declaração das operações matematicas
<OP_matematico>::= OP_mais | OP_menos | OP_vezes | OP_dividir;
//imprimir frases na pagina
<exibir>::= OP_exibir OP_paEsquerdo palavras OP_paDireito OP_virgula <maisexibir> |
OP_exibir OP_paEsquerdo OP_aspa palavras OP_aspa OP_paDireito OP_virgula <maisexibir> |
OP_exibir OP_paEsquerdo OP_aspa palavras OP_aspa OP_mais palavras OP_paDireito OP_virgula <maisexibir>;
<maisexibir>::= <exibir> | î;
```

Figura 3. Gramática do lado servidor

Nas Figuras 2 e 3, observam-se as delimitações “<” e “>” para os símbolos não terminais, bem como cada derivação é representada por “::=”, mas se palavra vazia tem-se “î”.

O compilador foi projetado para receber um programa escrito em português, que esteja de acordo com as regras sintáticas da gramática do lado cliente (Figura 2) ou do lado servidor (Figura 3). Por exemplo, tanto na gramática referente ao cliente da aplicação quanto na referente ao servidor, as derivações começam a partir da variável “<programa>” que é a raiz da arvore de derivação, partindo-se dessa variável o compilador executa as demais regras de derivação, como, por exemplo, a derivação necessária para gerar a declaração de uma caixa de texto no formulário é: “**<texto>**::=OP_texto palavras OP_virgula;”, para as demais derivações, por analogia, segue-se o mesmo processo.

O cliente é responsável pela criação de formulários e manipulação de documentos HTML, já o servidor aborda as questões de programação e manipula os

documentos JSP. Por fim, a função do compilador é analisar o código (reportando os possíveis erros ao usuário) e gerar o código Web para ser executado em um *browser*.

Conforme discutido, os analisadores léxicos e sintáticos são as duas primeiras partes do projeto de um compilador. A construção desses dois módulos sem a ajuda de uma ferramenta pode ocasionar eventuais erros, além de ser um esforço desnecessário, tendo em vista a existência de ferramentas para este fim.

Portanto, para o desenvolvimento dos módulos de análise léxica e análise sintática foi utilizado o GALS. Têm-se como principais características para a escolha do GALS: (i) é flexível na geração de código objeto em linguagens como C++, Delphi e Java; (ii) é uma ferramenta de código fonte aberto liberada sob a licença pública GNU (*General Public License*); e (iii) possui os analisadores tanto léxicos quanto sintáticos.

A partir das gramáticas propostas, o GALS gerou a codificação na linguagem Java, sendo esta responsável pela implementação dos analisadores léxicos e sintáticos. Posteriormente foi desenvolvido o algoritmo responsável pela geração de código objeto que poderá ser salvo em um arquivo de formato .jsp, o qual pode ser executado em um *browser* através de um servidor Web.

A ferramenta AAPW (Figura 4) possui IDE de programação e gerenciador do compilador. Desenvolvida em Java, sua IDE é composta de barra de ferramentas; das telas de edição do programa fonte e do código compilado (HTML/JSP); de um *browser* para visualizar as páginas desenvolvidas; e de um console que exibe mensagens de erros presentes no código pertencente ao programa fonte. Suas funcionalidades visam facilitar o processo de ensino-aprendizagem de desenvolvimento Web.

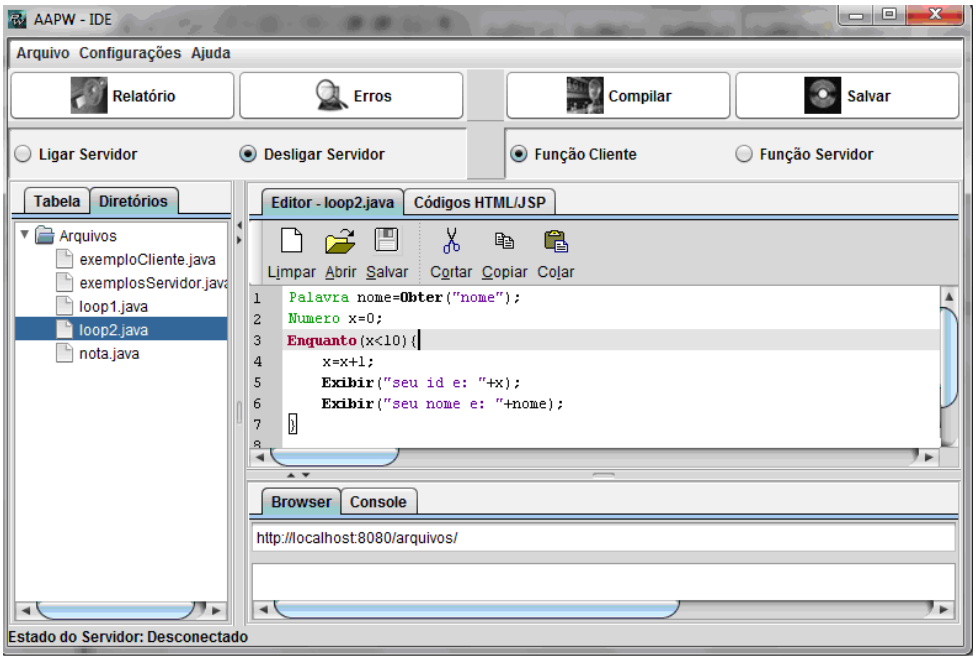

Figura 4. IDE da ferramenta AAPW

O desenvolvimento de um programa na ferramenta AAPW baseia-se em um fluxo básico, ilustrado na Figura 5, o qual inicialmente o usuário insere o código fonte no editor do IDE, o qual é analisado pela análise léxica da ferramenta. Se não ocorrerem erros léxicos a análise sintática é realizada executando a geração de código. O código

processado poderá ser salvo em arquivo de formato .jsp. A escolha de JSP está no fato da ferramenta AAPW utilizar o servidor Jakarta Tomcat[12].

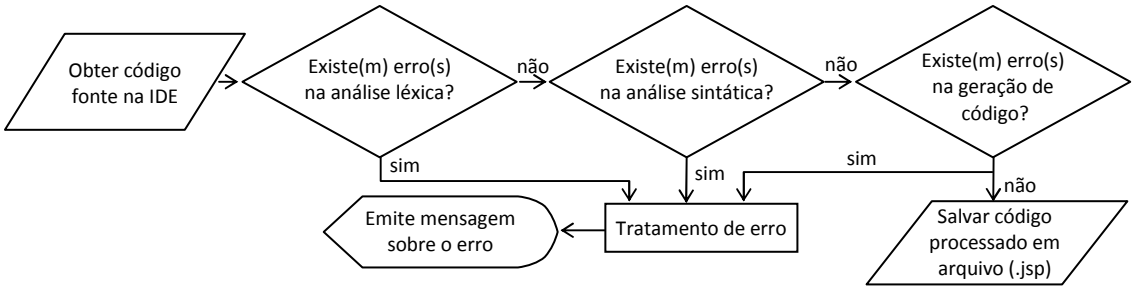

Figura 5. Fluxo básico para implementação de um programa em AAPW

A Figura 6 ilustra um programa desenvolvido na ferramenta AAPW.

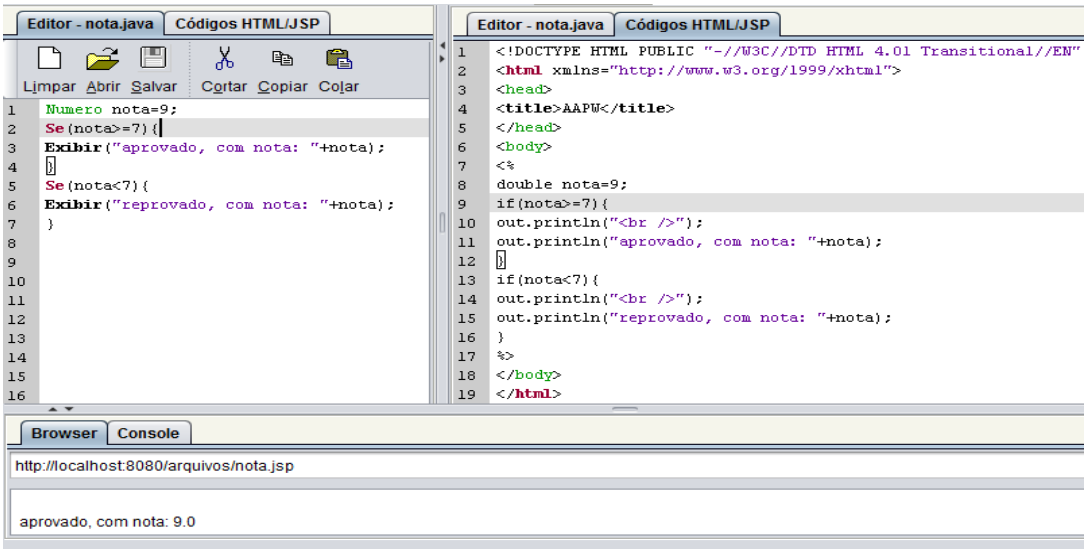

Figura 6. Exemplo de programa desenvolvido na ferramenta AAPW

A IDE de AAPW propicia ao usuário a visualização tanto do seu código construído em português (lado esquerdo da Figura 6), quanto à codificação gerada (lado direito da Figura 6), a qual pode ser interpretada em um *browser*. É possível observar o programa em execução no *browser* de AAPW (localizado na parte inferior da Figura 6).

5. Aplicação e validação da ferramenta

5.1. Técnicas de qualidade

Existem algumas soluções disponibilizadas para mensuração da qualidade de uso, como as metodologias de Gomes (2009), Santos (2007) e Oliveira (2006), além de questionários de satisfação, mas que não levam em consideração os aspectos pedagógicos nos softwares [Abreu 2010]. A Norma ISO 9241-11 [ISO 2002][Prumper 1999] define usabilidade como "a capacidade de um produto ser usado por usuários específicos para atingir objetivos específicos com eficácia e satisfação em um contexto especifico de uso".

[12] Disponível em <http://jakarta.apache.org/site/downloads/index.html>

Na avaliação de usabilidade, uma das classificações é quanto ao seu objetivo. Para esta classificação, existem três classes distintas de técnicas para avaliação de usabilidade, a saber: objetivas, prospectivas e preditivas [Cybis 2003].

Nas técnicas objetivas (ou interpretativas), o pesquisador direciona o uso do aplicativo aos usuários finais. Após monitoração de uso, os dados coletados devem ser interpretados. Dentre as metodologias está o teste Empírico Tradicional, que consiste na observação e monitoramento da interação do usuário com o sistema, em um ambiente parcialmente controlado, através da execução de uma bateria de atividades para as funcionalidades do aplicativo analisado [Prates e Barbosa 2003].

As técnicas prospectivas prevêem a prospecção das opiniões subjetivas dos usuários. Através de questionários ou entrevistas, avalia-se a satisfação do usuário em relação ao aplicativo e sua operação. Tem como dificuldade a taxa de devolução, cerca de 30% de retorno, mas a problemática é geralmente contornada se a metodologia for aplicada no próprio ambiente da interação com a aplicação. Os questionários de satisfação permitem levantar informações sobre experiências, opiniões e preferências. Tais dados são tão (ou mais) importantes quanto o próprio desempenho do aplicativo [Winckler 2001].

As técnicas preditivas (ou diagnósticas) são realizadas através de inspeções por especialistas em interface gráfica, no intuito de prever eventuais problemas. São mais viáveis de executar, pois não há esforço em recrutar usuários, entretanto sua eficiência depende, principalmente, da capacidade dos avaliadores [Abreu 2010].

Existe um atrativo na utilização de questionário como ferramenta de avaliação: os achados produzidos pela investigação, independentemente dos sistemas, usuários e tarefas, são passíveis de descrição e análise estatística [Nielsen 1994]. Isto torna tal técnica uma ferramenta de sondagem de aplicação rápida, reduzindo os custos envolvidos com a administração e computação dos resultados.

5.2. Validação da ferramenta

Para a avaliação da usabilidade da ferramenta AAPW, foi dada ênfase à avaliação objetiva e prospectiva por meio de questionários, uma vez que sua adoção permite a tabulação automatizada dos dados.

A escolha da técnica objetiva se deve ao fato de permitir simular o uso do aplicativo com os usuários finais através do teste empírico tradicional, através do monitoramento da interação do usuário com a aplicação em uma bateria de atividades que utilizam as funcionalidades da ferramenta AAPW. Já a escolha da avaliação de usabilidade prospectiva se deve ao fato da técnica permitir a aplicação de questionários de satisfação de usuário, consequentemente pode-se avaliar a interação que o usuário teve com a ferramenta na aplicação da primeira técnica.

O questionário[13] visa avaliar o grau de usabilidade da ferramenta AAPW e foi desenvolvido a partir de SAUSP (Sistema Avaliador de Usabilidade em *Softwares* Pedagógicos) [Abreu 2010] e ISONORM 9241/10 [Prümper 1999].

[13] Disponível em <https://docs.google.com/spreadsheet/viewform?fromEmail=true&formkey=dGxjYU4yUVJYTnpJZDdkZFZpaVAtU1E6MQ>

Para a avaliação participaram 50 alunos do curso de Bacharelado em Sistemas de Informação (BSI) da Unidade Acadêmica de Serra Talhada (UAST) da Universidade Federal Rural de Pernambuco (UFRPE), uma das mais tradicionais IFES do Nordeste.

Antes da aplicação dos exercícios, ofertou-se um minicurso para todos os alunos. Abordaram-se conceitos básicos de HTML e JSP, mas sem a utilização de ferramentas. Os alunos foram divididos aleatoriamente em dois grupos: (i) 25 indivíduos que usariam a ferramenta APPW para solucionar problemas; e (ii) 25 indivíduos que solucionariam problemas por abordagens tradicionais.Vale ressaltar que em ambos os grupos, os alunos não possuíam noções sobre programação Web, apenas noções sobre programação básica (definições de variáveis, comandos de repetição e condicionais, etc.).

Foram aplicados exercícios entre os alunos nos laboratórios de computação do curso de BSI da UAST/UFRPE. Os exercícios envolviam soluções para declarações de variáveis, estruturas de controle, estruturas de decisão e formulários Web, a saber: (1) um programa que receba um nome e o envia para outra página, na qual será impresso dez vezes; (2) um programa que receba a nota de duas provas. Este deve imprimir aprovado, se a média for maior ou igual a sete, ou reprovado, caso contrário; (3) criação de um formulário com campos para nome, CPF, sexo, cidade e comentário. Em seguida exibe-se o conteúdo desses dados em outra página.

Aos alunos do grupo (i) explanou-se a ferramenta AAPW com a demonstração de suas funcionalidades. Logo após, foi solicitado aos alunos que resolvessem os exercícios propostos. Uma vez finalizados, aplicou-se o questionário para coletar dados referentes à usabilidade da ferramenta.

No grupo (i), notou-se um bom desempenho dos alunos com a resolução da maioria dos problemas dos exercícios propostos, enquanto que o grupo (ii) apresentou notória dificuldade em solucionar tais problemas. Com base nos dados coletados do questionário pode-se notar que a ferramenta teve grande aceitação para os alunos contribuindo para o aprendizado. Para a análise adotou-se o teste de *Qui-quadrado* a partir da frequência das respostas dos alunos para o questionário. O Valor obtido foi de X^2=13,09, fato que ao nível de significância 0,05 com 4 graus de liberdade, evidencia que o grau de usabilidade oferecido pela ferramenta fornece contribuições significantes para o processo de ensino-aprendizagem de desenvolvimento Web.

5. Conclusão e contribuições

No processo ensino-aprendizagem de desenvolvimento Web é comum observar diversas dificuldades entre os alunos no entendimento da programação envolvida, tendo em vista que muitas das linguagens para este fim possuem elevado número de comandos, como, por exemplo, das *tags* em documentos HTML.

Diante da problemática, este trabalho tem como principal contribuição à criação de uma ferramenta que poderá ser utilizada no processo de ensino-aprendizagem por professores e alunos em disciplinas que introduzem a programação para desenvolvimento Web. Facilita o entendimento do processo de codificação e a visão geral da programação cliente-servidor, por meio de comandos em português que elimina a necessidade de inserção de *tags*. Também é possível produzir código funcional, ou seja, codificação que pode ser publicada em qualquer servidor Web.

Como trabalho futuro, pretende-se desenvolver novas funcionalidades envolvendo a geração de código alvo para outras linguagens, a exemplo de PHP, bem como da possibilidade de conexão com banco de dados. Além disto, pretende-se realizar outros experimentos visando identificar eventuais falhas, e que complementarão o estudo de caso para validação da ferramenta

Referências

Abreu, A. C. B. (2010) Avaliação de usabilidade em softwares educativos. Dissertação de Mestrado em Computação Aplicada, Universidade Estadual do Ceará, 109 p.

Aho, A. V., Sethi, R. e Ullman, J. D. (1995) Compiladores: Princípios, Técnicas e Ferramentas, Rio de Janeiro: LTC.

Almeida, E. S., Herreral, J. D., Filho, L. J. S., Almeida, H. O., Costa, E. B., Vieira, B. L. e Melo, M. D. (2004) "Um Ambiente Integrado para auxílio ao Ensino de Ciência da Computação". In: *Revista Digital da CVA*, v. 2, n. 8.

Cybis, W. A. (2003) Engenharia de Usabilidade: Uma Abordagem Ergonômica, http://www.unoescsmo.edu.br/poscomp/cybis/Apostila_v51.pdf.

Gesser, C. E. (2003) GALS - Gerador de Analisadores Léxicos e Sintáticos. UFSC.

Gomes, A. S. (2009) FAVIHC – Framework de Avaliação da Interação Humano-Computador. Dissertação apresentada ao Curso de Mestrado em Informática Aplicada da Universidade de Fortaleza, 147 p.

ISO (2002). ISO 9241-11. Requisitos Ergonômicos para Trabalho de Escritório com Computadores: Parte 11 – Orientações sobre Usabilidade.

Lesk, M. E. (1975). Lex – a lexical analyzer generator. Computing Science Technical Report 39, AT&T Bell Laboratories. Murray Hill, N. J.

Menezes, P. B. (2005) Linguagens formais e Autômatos. 5ª ed., Porto Alegre: Bookman.

Nielsen, J. (1994) Ten Usability Heuristics, http://www.useit.com/papers/heuristic/heuristic_list.html.

Oliveira Junior, J. A. G. (2006) Apoio à Avaliação de Usabilidade na Web – desenvolvimento do USEWEB. Dissertação. Instituto de Computação, UECG. 116 p.

Prates, R. O. e Barbosa, S. D. J. (2003) Avaliação de Interfaces de Usuário – Conceitos e Métodos, http://www.inf.puc-rio.br/~inf1403/docs/JAI2003_PratesBarbosa_avaliacao.pdf.

Price, A. M. A. e Toscani, S. S. (2005) Implementação de Linguagens de Programação: Compiladores, 3ª ed., Porto Alegre: Bookman.

Prumper, P. (2009) Test it: ISONORM 9241/10. In *Proceedings of HCI International*, Munich, pages 22-17. Lawrence Erlbaum, Mahwah, NK, USA.

Santos, R. C. (2007) Desenvolvimento de uma Metodologia para Avaliação de usabilidade de sistemas utilizando à lógica Fuzzy baseado na ISO. Dissertação - Ibmec. 115 p.

Sebesta, R. W. (2003) Conceitos de linguagens de Programação. Santos, J. C. B. (Trad.), 5ª ed., Porto Alegre: Bookman.

Winckler, M. (2001) Avaliação de Usabilidade de Websites. In *IV Workshop sobre Fatores Humanos em Sistemas de Computação*, Florianópolis.