RBECT
Revista Brasileira de Ensino de Ciência e Tecnologia

Revista Brasileira de Ensino de Ciência e Tecnologia
ISSN: 1982-873X

https://periodicos.utfpr.edu.br/rbect

Utilização de um rastreador ocular para análise de estratégias de leitura de programas de computador por discentes

Bianca Alessandra Visineski Alberton
bi.alberton@gmail.com
0000-0001-7680-7300
Universidade Tecnológica Federal do Paraná – UTFPR – DAELN – Campus Curitiba, Paraná, Brasil.

Diego Gabriel Lee
diegolee7@gmail.com
0000-0001-7074-6332
Universidade Tecnológica Federal do Paraná – UTFPR – DAELN – Campus Curitiba, Paraná, Brasil.

Maiko Min Ian Lie
minian.lie@gmail.com
0000-0003-4190-2556
Universidade Tecnológica Federal do Paraná – UTFPR – CPGEI – Campus Curitiba, Paraná, Brasil.

Humberto Remigio Gamba
humberto@utfpr.edu.br
0000-0003-3210-2725
Universidade Tecnológica Federal do Paraná – UTFPR – CPGEI – Campus Curitiba, Paraná, Brasil.

Gustavo Benvenutti Borba
gustavobborba@utfpr.edu.br
0000-0001-7412-2412
Universidade Tecnológica Federal do Paraná – UTFPR – PPGEB – Campus Curitiba, Paraná, Brasil.

RESUMO

O presente artigo relata a elaboração e execução de um experimento para analisar as estratégias empregadas por discentes na leitura de programas escritos na linguagem C. Foi utilizado um rastreador ocular para determinar as regiões observadas no código e, a partir destas regiões, quais as estratégias de leitura utilizadas. Para determinar estas áreas, foram capturadas as fixações em cada linha do programa. Por meio da análise dos movimentos oculares foi possível verificar a ocorrência do padrão de primeira leitura descrito na literatura. A divisão do código em blocos semânticos revelou que os estudantes passam a maior parte do tempo no bloco de processamento e que as transições entre os blocos ocorrem de forma bastante dinâmica. Os resultados indicam que variações na sintaxe das linguagens de programação utilizadas são um aspecto tão importante quanto o tempo de experiência na depuração de defeitos lógicos simples em códigos-fonte de programas de computador.

PALAVRAS-CHAVE: Compreensão de programas de computador. Rastreador ocular. Fixações.

1. INTRODUÇÃO

Conteúdo básico em todas as engenharias, a programação de computadores é uma disciplina que exige muito raciocínio lógico e é considerada por muitos estudantes como difícil de aprender e utilizar (TAN, 2009).

De modo a auxiliar o aprendizado deste tópico, foram criadas linguagens de programação educacionais como o *LOGO* (DU BOULAY et al., 1981) e o *Scratch* (MALONEY et al., 2009, p.60). Entretanto, para tópicos mais avançados, como identificação de erros em códigos ou análise de programas complexos, tais linguagens não são indicadas, visto que enfatizam somente o aprendizado de lógica básica de programação.

De acordo com Bednarik et al. (2006), o aprendizado de novas estratégias de programação é possível a partir da compreensão das estratégias utilizadas por programadores experientes. Assim, entender os processos cognitivos referentes à leitura e produção de código tem muito a contribuir com o ensino de programação.

Nesse sentido, o presente trabalho apresenta um experimento para analisar o processo de leitura de programas de computador (códigos-fonte) por meio da utilização de um rastreador ocular (*eye tracker*), com o objetivo de compreender as estratégias empregadas neste processo. O experimento contou com a participação de 34 voluntários, todos discentes dos cursos de bacharelado em Engenharia de Computação e Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná - Campus Curitiba. Este documento está organizado como segue: na Seção 2 são relatados os trabalhos relacionados, de modo a verificar o procedimento utilizado por outros autores para analisar os dados oculares durante uma leitura de programas. Na Seção 3 descreve-se a metodologia adotada no experimento proposto. Na Seção 4 são relatados os resultados obtidos. Por fim, na Seção 5, são apresentadas as considerações finais.

2. TRABALHOS RELACIONADOS

Os primeiros estudos realizados utilizavam variações da técnica *thinking-aloud* (BEDNARIK, 2007). Por meio desta técnica, o programador explica o seu raciocínio enquanto lê um código de computador. Este método possui duas desvantagens principais: para aplicá-lo, o programador precisa passar por um treinamento para se acostumar a falar o que pensa; e ao falar o que pensa, o programador se desconcentra da tarefa principal de analisar o programa (BEDNARIK, 2007).

Estudos mais recentes empregam o rastreador ocular como forma de substituir o método *thinking-aloud*, uma vez que este equipamento não interfere na atividade realizada pelo programador. De acordo com Rayner (1998), o indivíduo conduz o seu olhar para as mesmas regiões que despertam a sua atenção, especialmente em tarefas que exigem a compreensão de conteúdos complexos. De modo complementar, Just e Carpenter (1980, p. 330) afirmam que a duração do olhar em uma determinada região é proporcional ao tempo que o indivíduo leva para compreender o que está sendo observado. Desta forma, os movimentos oculares fornecem informações relevantes sobre quais são as áreas que o programador passa mais tempo processando.

Uwano et al. (2006) utilizaram o rastreador ocular para comparar as estratégias utilizadas na busca de defeitos (erros) lógicos em códigos na linguagem

C. Neste experimento, seis programadores analisaram seis programas, os quais possuíam um único defeito lógico, por até cinco minutos. Entre os resultados obtidos, Uwano et al. (2006) afirmaram que programadores que passam mais tempo realizando uma primeira leitura do programa (*first scan*) são capazes de encontrar o defeito mais rápido. Além disso, relatam ainda a existência de dois outros padrões: releitura de declaração de variável (*retrace declaration pattern*), por meio da qual o programador olha a declaração de uma variável pouco depois dela ser utilizada pela primeira vez; e releitura da última referência (*retrace reference pattern*), que é encontrada quando o programador lê uma linha *L* do código que utiliza um conjunto de variáveis *v* e, logo em seguida, relê as últimas linhas onde estas variáveis *v* foram empregadas.

De modo a analisar os dados oculares com uma amostra mais significativa, Sharif et al. (2012) reproduziram este experimento com quinze programadores, mas apresentando quatro dos códigos propostos por Uwano et al. (2006). Sharif et al. (2012) também verificaram os efeitos da primeira leitura do programa, mas não encontraram ocorrências significativas dos padrões de releitura de declaração de variável e releitura da última referência.

Um dos trabalhos mais recentes nesta área foi a proposta de uma metodologia de análise de dados oculares na leitura de programas de computadores (BUSJAHN et al., 2014). Por meio desta metodologia, a leitura de um programa é realizada não só em função de cada uma de suas linhas, mas também em função dos blocos e sub-blocos nas quais elas estão inseridas, de certos padrões de leitura e das estratégias utilizadas. Dentre os padrões de leitura, destacam-se a leitura na mesma ordem de execução do programa (*jump control*) e a leitura linear de uma parte do código (*linear scan*).

3. METODOLOGIA

Nesta seção são relatadas a captura dos dados de movimento ocular por meio do equipamento, a realização do experimento e a metodologia de análise dos dados obtidos.

3.1. Movimentos oculares

A aquisição de informações visuais por meio da leitura de programas é realizada principalmente a partir de dois movimentos oculares: as fixações e os movimentos sacádicos (BUSJAHN et al., 2014).

As fixações consistem em pequenos movimentos oculares em torno de um ponto, os quais possibilitam a estabilização da retina (DUCHOWSKI, 2007). Eles duram até 600 ms e abrangem até 5° do ângulo visual. Considera-se uma fixação quando mais de 5 pontos são detectados em um raio de 7 mm (ZÜLCH; STOWASSER, 2003). Os movimentos sacádicos são movimentos rápidos cuja função é reposicionar a fóvea. Eles possuem duração de 10 a 100 ms e ocorrem sempre entre duas fixações (DUCHOWSKI, 2007). De acordo com Fuchs (2012), pouco ou nenhum processamento é realizado durante os movimentos sacádicos.

Os rastreadores oculares atuais são capazes de gravar, de forma não invasiva, os movimentos oculares de um indivíduo, permitindo a liberdade de movimento

da cabeça. Estes instrumentos utilizam fontes de luz infravermelha para projetar padrões de luminosidade nos olhos do sujeito e, a partir de câmeras, capturar a luz refletida pelos olhos. Com isso, é possível determinar, em tempo real, os pontos na tela do computador para os quais direcionou-se o olhar.

3.2. Experimento

O experimento, aprovado pelo Comitê de Ética em Pesquisa com Seres Humanos da UTFPR conforme o parecer 1.483.726, foi realizado com o rastreador ocular RED 500 do fabricante alemão SMI (SMI, 2016a). Esse equipamento é capaz de recolher dados binoculares e possui exatidão de 0,4° no rastreamento do olhar. Ele é composto pelo conjunto rastreador-monitor e um notebook, no qual há softwares da SMI para gravação dos movimentos oculares, elaboração do experimento e processamento dos dados obtidos. O rastreador localiza-se abaixo do monitor e foi posicionado a uma distância entre 60 e 80 cm do participante, o qual possui liberdade para movimentar-se. O monitor possui 22” e a captura dos dados oculares foi feita a uma frequência de 500 Hz.

3.2.1. Seleção de voluntários

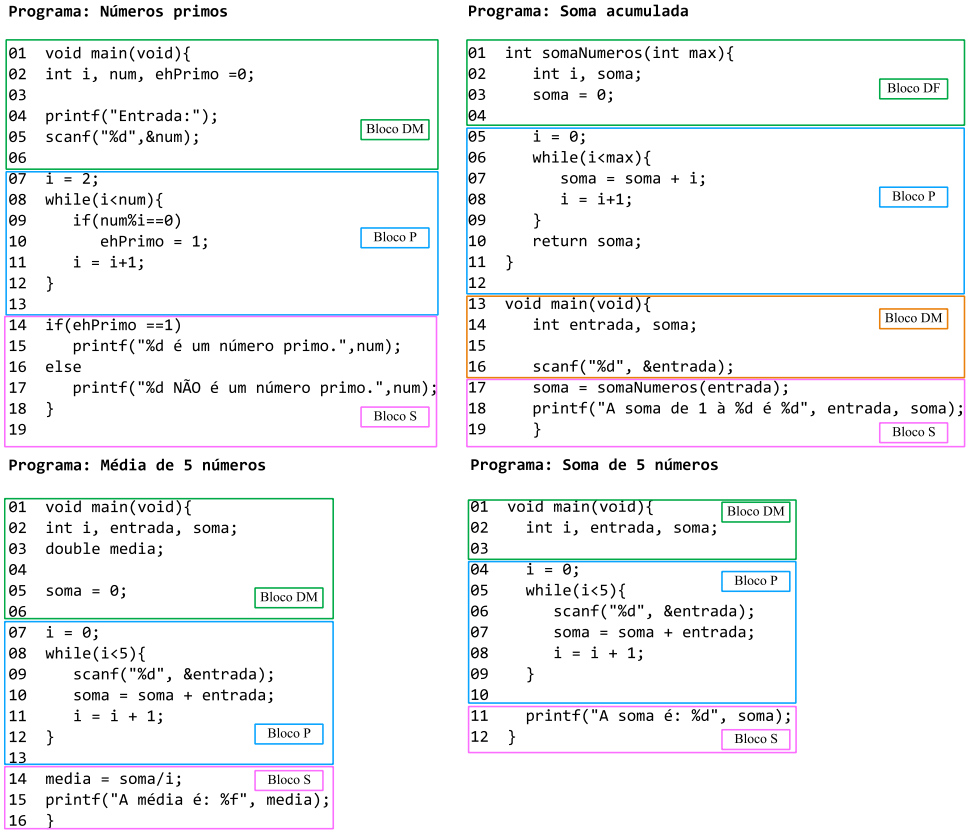
Para participar do experimento foram convidados alunos dos cursos de Bacharelado em Engenharia de Computação e de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná (UTFPR). Participaram tanto alunos com visão normal quanto aqueles que utilizam óculos ou lentes de contato, visto que o rastreador ocular RED 500 consegue capturar os dados oculares nos três casos. Os únicos critérios para seleção de voluntários para o experimento foram: idade acima de 18 anos e ter sido aprovado na disciplina de Programação 1.

Antes de iniciar o experimento, os alunos foram instruídos sobre o procedimento a ser seguido. Eles foram encorajados a evitar movimentos amplos durante a calibração e a leitura do programa. Em seguida, os participantes leram e preencheram o termo de consentimento livre e esclarecido. No total, participaram 34 voluntários, dos quais 17 estavam cursando até o 4º período de seu curso e 17 estavam cursando a partir do 7º período – os cursos envolvidos têm duração total de 10 períodos. Os participantes possuíam idades entre 18 e 31 anos (média de 22 anos). Apenas dois dos alunos já haviam exercido atividades profissionais utilizando a linguagem C. Ambos estavam cursando a partir do 7º período e já haviam programado na linguagem durante o estágio.

3.2.2. Programas analisados

Com o objetivo de verificar os resultados obtidos por Uwano et al. (2006) e Sharif et al. (2012), foram escolhidos quatro programas na linguagem C sugeridos por estes autores, cada um com um defeito (erro) lógico inserido. A Figura 1 exibe os quatro programas, bem como uma divisão do código em blocos semânticos. A divisão compreende três blocos: declaração de variáveis, processamento dos dados (P) e saída do programa (S). O bloco de declaração de variáveis pode ocorrer apenas na função *main* (DM) ou em outras funções (DF). Os alunos não possuíam acesso a esta divisão em blocos.

Figura 1 – Programas analisados pelos alunos e sua divisão em blocos. DM: declaração de variáveis na função main; DF: declaração de variáveis em outras funções; P: processamento dos dados; S: saída do programa.



(Fonte: Adaptado de Uwano et al. (2006))

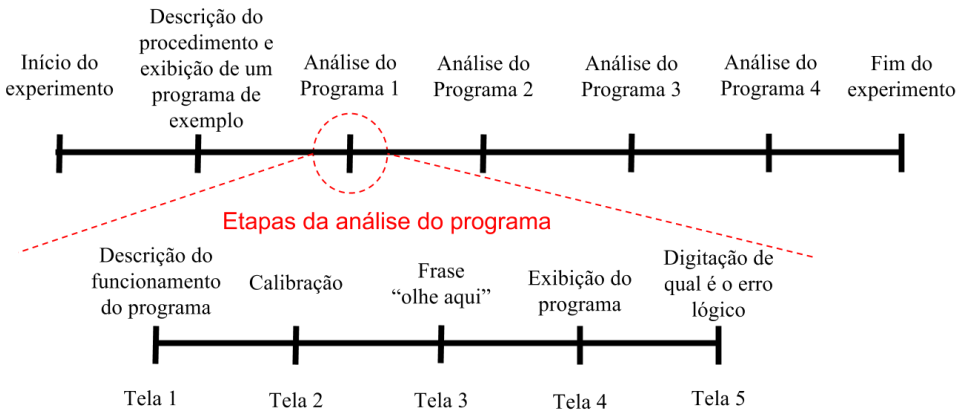
Os defeitos inseridos nos códigos são os seguintes: no programa “Números primos” a lógica de atribuição se um número é primo ou não está invertida (linhas 2, 10); no programa “Soma acumulada” o último número não é contabilizado na soma (linha 6); no programa “Soma de 5 números” a variável soma não é inicializada (linha 2); e no programa “Média de 5 números” a divisão de dois inteiros resulta em um número inteiro, retornando a média incorreta (linha 14).

Antes dos voluntários iniciarem a análise dos programas, estes foram instruídos em relação ao procedimento a ser seguido durante a coleta dos dados e observaram um programa exemplo, conforme descrito na Figura 2. Os códigos foram exibidos no monitor de 22” utilizando fonte Arial preta de tamanho 24 pt. O fundo da tela era cinza e cada programa possuía entre 12 e 19 linhas, não sendo necessário utilizar a barra de rolagem. A ordem de análise dos códigos foi aleatória e organizada por meio do software *Experiment Center 3.6* (SMI, 2016b). Este *software* também realiza a captura dos dados oculares e é fornecido pelo fabricante SMI. A partir dele é possível elaborar várias telas para exibição de textos e imagens.

O procedimento para a análise de cada um dos quatro códigos foi realizado em cinco etapas, conforme exibido no detalhe expandido da Figura 2. Primeiramente foi exibida uma tela com a descrição do programa com exemplos de entrada e saída, de modo que o participante conseguisse entender a finalidade do código a ser analisado. Durante esta etapa os alunos poderiam fazer perguntas

e tirar dúvidas sobre o funcionamento do programa. Em seguida, na segunda etapa, foi realizada a calibração do equipamento utilizando 13 pontos na tela.

Figura 2 – Procedimento utilizado na realização do experimento.



A terceira etapa consistiu em uma tela com a frase "olhe aqui", a qual estava programada para passar para a quarta etapa após o participante olhar para a frase por 1 s. Tal tela teve como objetivo garantir que o aluno começaria a ler o código com os olhos fixados em uma região próxima à primeira linha. Na quarta etapa, o programa foi exibido para o voluntário por até cinco minutos. Caso o aluno encontrasse o erro lógico antes do tempo limite, poderia passar para a quinta e última etapa apertando a tecla de espaço. Por fim, o aluno deveria digitar na última tela qual era o defeito lógico (erro) do programa e em qual linha ele estava localizado. Os dados oculares do participante foram registrados apenas durante a etapa de calibração e de leitura do programa.

A análise dos dados oculares na leitura do código foi realizada em função das linhas lidas, conforme proposto por Uwano et al. (2006), e também em função dos blocos semânticos, conforme proposto por Busjahn et al. (2014).

4. RESULTADOS

A leitura de cada um dos quatro programas feita pelos 34 participantes resultou em 136 amostras de dados oculares para análise. Destas 136 foram descartadas 37 amostras (27 %), devido a problemas na captura dos movimentos oculares dos participantes. De acordo com Goldberg e Wichansky (2003), cerca de 20 % dos participantes encontram dificuldades com a calibração ou o sistema perderá a localização de seus olhos durante o experimento.

No experimento realizado, observou-se que 24 leituras de código entre as 37 amostras descartadas ocorreram entre os alunos dos primeiros períodos. Eles demonstraram ansiedade e se movimentaram bastante. Embora tenham sido instruídos que nenhuma pergunta seria respondida durante a leitura do programa, alguns desviaram o olhar da tela para fazer perguntas à operadora do equipamento. Todos estes fatos contribuíram para o descarte das amostras.

Inicialmente foi realizada uma comparação entre as diferenças no padrão de leitura de códigos separando os alunos em dois grupos: um com os alunos que estavam cursando até 4º período (grupo A) e outro com os alunos a partir do 7º

período (grupo B). A Tabela 1 exibe os dados obtidos para ambos grupos, referentes ao tempo médio para concluir onde o defeito lógico estava e a resposta fornecida pelo participante. Embora alguns alunos tenham concluído rapidamente onde o erro estava localizado, isso não significa que ele realmente o encontrou, pois a sua hipótese poderia estar incorreta.

Tabela 1 - Tempo médio para encontrar o erro lógico e quantidade de respostas corretas fornecidas pelos participantes dos grupos A (alunos cursando até o 4º período) e B (alunos cursando a partir do 7º período).

	Grupo de participantes	**Programa**			
		Números primos	**Soma acumulada**	**Soma de 5 números**	**Média de 5 números**
Tempo médio para concluir a resposta (s)	A	158	117	58	191
	B	165	124	88	113
Respostas corretas (%)	A	42	73	100	22
	B	13	93	57	7

Como é possível observar na Tabela 1, os alunos do grupo A acertaram mais respostas que os alunos do grupo B na maioria dos programas e, com exceção do programa “Média de 5 números”, também chegaram a uma conclusão mais rapidamente. Este resultado deve-se principalmente ao fato de os alunos do grupo A estarem ativamente programando na linguagem C durante os 3 primeiros períodos do curso e os alunos do grupo B utilizarem mais outras linguagens nos períodos finais dos cursos, tais como Java e MATLAB. Isto ficou bastante evidente quando, após realizarem o experimento, muitos dos alunos dos períodos mais avançados disseram, espontaneamente, que não programavam na linguagem C já fazia algum tempo. Isso sugere fortemente que a sintaxe da linguagem de programação envolvida tem um papel tão significativo quanto o tempo de experiência do programador na análise de problemas lógicos simples.

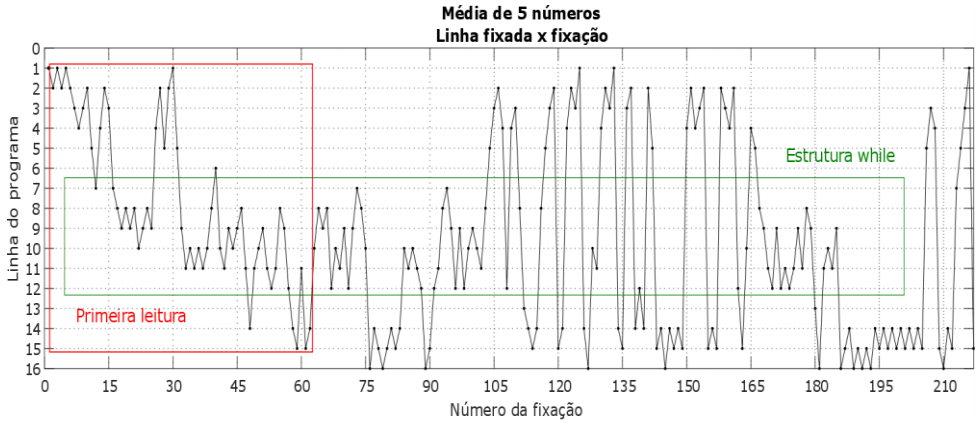
Além disso, alguns dos erros lógicos inseridos nos códigos não ocorre em outras linguagens com tanta frequência, como a divisão de dois números inteiros no programa “Média de 5 números”. Em linguagens como MATLAB, por exemplo, não é necessária a declaração de variáveis e o tipo (classe) padrão para a representação numérica é o de dupla precisão (*double*), o que não geraria um erro. Isso evidencia a significância da tipagem (o rigor da linguagem em relação à utilização dos tipos de dados) da linguagem de programação utilizada na eficiência da depuração de problemas lógicos simples.

No total foram obtidas 40 respostas corretas entre os 99 dados de programas analisados. Acredita-se que a taxa de acerto poderia ser melhorada se os alunos tivessem acesso a um editor de texto e compilador para executar o programa, pois eles poderiam corrigir o erro e verificar se a sua hipótese estava correta antes de escrevê-la.

4.1 Primeira leitura do programa

Conforme descrito por Uwano et al. (2006), a primeira leitura é realizada quando o programador lê 80% das linhas do código. Assim, verificou-se o tempo gasto pelos participantes na realização da primeira leitura. A Figura 3 exibe a ordem das linhas fixadas durante a análise do programa "Média de 5 números" para um participante. Na figura também é possível observar a ocorrência da primeira leitura.

Figura 3 – Primeira leitura realizada por um dos participantes no programa "Média de 5 números".



Apenas um dos participantes não apresentou o padrão de primeira leitura. Todos os outros apresentaram este padrão que, em média, corresponde a 32% do tempo total empregado na compreensão do código.

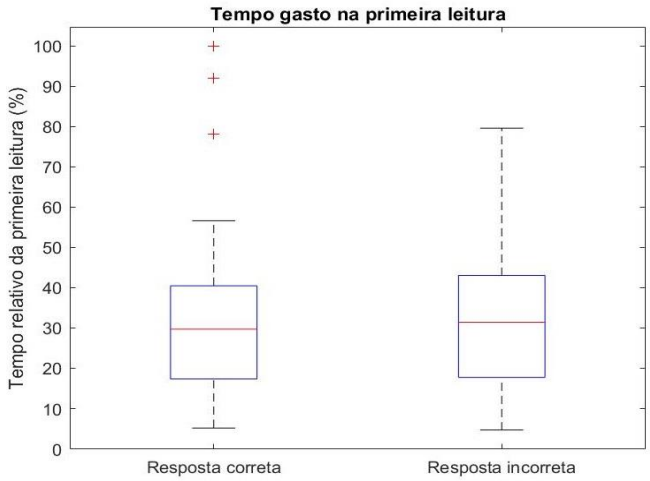
O tempo relativo empregado na primeira leitura (tempo em relação ao utilizado para encontrar o erro) não difere significativamente entre os participantes que acertaram e aqueles que erraram qual era o defeito lógico inserido no código, conforme é possível observar na Figura 4. Foram considerados como resposta incorreta também os casos em que o aluno não conseguiu encontrar o erro. Optou-se por abordar a primeira leitura desta forma para possibilitar o agrupamento dos dados dos quatro códigos analisados.

4.2 Análise da leitura por blocos semânticos

Além da análise da leitura do programa por meio das linhas fixadas, também foram analisadas as fixações em cada bloco semântico (descritos na Figura 1). Conforme explicitado na Seção 3.2, a divisão foi feita em três tipos de blocos: declaração de variáveis, processamento dos dados e saída do programa. Optou-se por incluir no bloco de processamento a atribuição da variável *i*, uma vez que ele está intimamente conectado à estrutura *while*. Embora alguns blocos ainda pudessem ser subdivididos (i.e. o bloco de processamento do programa "Números primos" inclui um sub-bloco com a estrutura condicional *if*), optou-se por manter a análise em função destes três blocos por representarem três etapas bem distintas de um programa: a entrada de dados, o seu processamento e a saída. O único programa que possui quatro blocos é o da soma acumulada, pois, como este

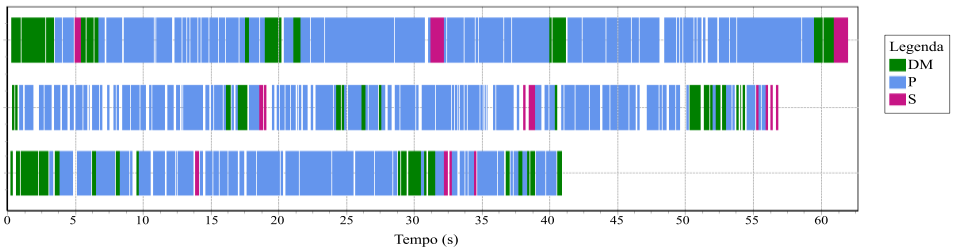
código possui uma função, as variáveis são inicializadas tanto na função principal do programa (DM), quanto dentro da função *somaNumeros* (DF).

Figura 4 – Comparação entre os tempos relativos gastos na primeira leitura dos participantes que forneceram a reposta correta e a incorreta para o problema dado.



A Figura 5 exibe a sequência de leitura dos três blocos do programa “Soma de 5 números” realizada por três participantes. Os espaços em branco na sequência de leitura indicam a ocorrência de movimentos sacádicos, a leitura de regiões fora do código ou piscadas realizadas pelos alunos.

Figura 5 – Sequência de leitura de três participantes para o programa “Soma de 5 números” em função dos blocos declaração de variáveis na função *main* (DM), processamento (P) e saída de dados (S).

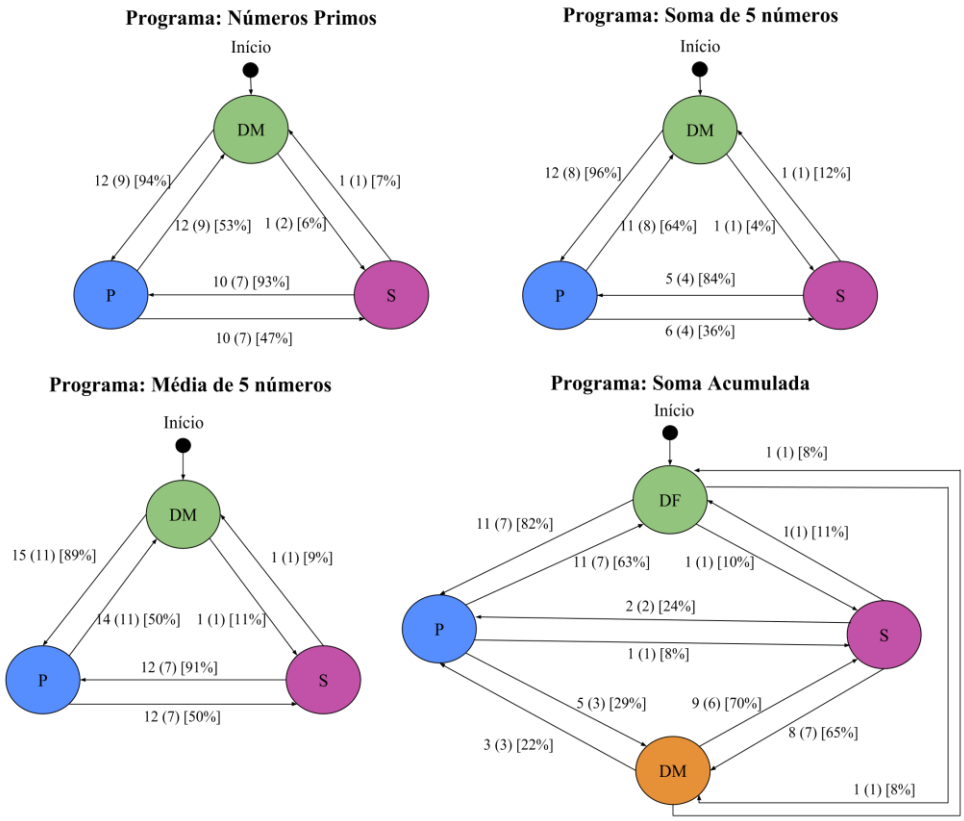


Por meio desta separação, verificou-se que em todos os programas os alunos passaram mais tempo analisando os blocos de processamentos de dados do que os outros. Em média, o tempo utilizado no bloco de processamento foi de 51% para o programa “Soma Acumulada”, 66% para o “Média de 5 números” e 73% para os programas “Números primo” e “Soma de 5 números”. Este tempo deve-se ao fato de que, para encontrar o defeito lógico, o aluno lê o programa na mesma sequência realizada pelo computador. Assim, ele realiza a leitura do código presente na estrutura *while* durante aproximadamente o mesmo número de vezes que o computador a executaria. Este processo ocupa a maior parte do tempo de leitura. Isso explica também porque o tempo de leitura é maior neste bloco até mesmo no programa “Soma de 5 números”, cujo defeito lógico está no bloco de declaração de variáveis na função *main* (DM).

Observou-se também que, apesar de os alunos simularem a sequência de execução do programa realizada pelo computador, essa sequência não é exatamente linear, pois muitas vezes eles voltam a observar o bloco de declaração de variáveis e de saída, conforme ilustrado no exemplo da Figura 3, na região destacada em verde. Este comportamento também foi verificado por Uwano et al. (2006).

De modo a analisar o fluxo de leitura do código dos participantes, gerou-se um diagrama de transição entre os blocos para cada programa (Figura 6). Nestes diagramas, os nós indicam os blocos do programa, enquanto as arestas indicam o número médio de transições realizadas e o desvio padrão. Também foi incluída a taxa média (porcentagem) com que estas transições ocorrem em relação ao número de transições de cada nó.

Figura 6 – Diagramas de transição das leituras dos programas. Cada nó indica um bloco do programa – declaração de variáveis na função principal (DM), declaração de variáveis em outras funções (DF), processamento dos dados (P) e saída do programa (S). As arestas indicam o número de transições realizadas no formato *média (desvio padrão) [porcentagem]*.



Verificou-se que, para os programas “Números primos”, “Soma de 5 números” e “Média de 5 Números” os alunos realizam poucas transições entre os blocos de declaração de variáveis e de saída (6%, 4% e 11% respectivamente), se comparadas com as transições entre o bloco de declaração e de processamento (94%, 96% e 89% respectivamente). Este resultado era esperado, visto que as variáveis possuem seus valores modificados durante o bloco de processamento. Assim, é

pouco provável que o aluno leia o bloco de declaração e passe para o de saída sem passar pelo de processamento e vice-versa.

No programa "Soma acumulada" há outra função além da função *main*, denominada *somaNumeros*. A chamada da função *somaNumeros* a partir de *main* faz parte do bloco de saída, pois é esta chamada que define a saída do programa. Neste caso, o aluno passa da leitura da linha contendo a atribuição da variável de entrada (bloco DM) para a chamada da função *somaNumeros* (bloco S), a qual está na linha seguinte, em 70 % das transições que saem do bloco DM.

Com relação às transições realizadas a partir do bloco de processamento, verificou-se que para os programas "Soma de 5 números" e "Média de 5 Números" as transições para os blocos de declaração de variáveis e saída ocorrem de forma equilibrada. Entretanto, para o programa "Soma de 5 números", as transições para o bloco de declaração ocorrem quase duas vezes mais do que para o de saída. Este efeito pode ser explicado pelo fato de o defeito lógico do programa ser a falta da declaração da variável *soma*.

No programa "Soma acumulada", é importante notar que o bloco de processamento encontra-se dentro da função *somaNumeros*. Assim, conforme esperado (UWANO, 2006), a maior parte (63%) das transições de saída deste bloco direcionaram-se para o bloco de declaração de variáveis da própria função (DF).

5. CONSIDERAÇÕES FINAIS

Por meio da análise dos movimentos oculares foi possível observar que inicialmente os alunos realizam uma primeira leitura do programa (lêem 80% das linhas do código), conforme proposto por Uwano et al. (2006). Apenas um dos participantes não apresentou este padrão de leitura. O tempo relativo empregado na primeira leitura (tempo em relação ao utilizado para encontrar o erro) não difere significativamente entre os participantes que acertaram e aqueles que erraram qual era o defeito lógico inserido no código, conforme é possível observar na Figura 4.

A comparação entre discentes dos primeiros e dos últimos períodos revelou que os graduandos de engenharia encontram mais dificuldades na identificação dos erros lógicos do que os iniciantes nos cursos. Isto se deve, em grande parte, ao fato dos alunos dos períodos avançados não programarem em C com a mesma regularidade que os alunos dos períodos iniciais. Assim, uma das principais contribuições deste trabalho é a verificação de que a sintaxe das linguagens de programação, mesmo que muito semelhantes, tem um impacto significativo no tempo de depuração de falhas lógicas simples. Apesar de todos os voluntários conhecerem a linguagem de programação C, verificou-se que aqueles que programaram predominantemente em outras linguagens no período de alguns meses antes do experimento apresentaram menor desempenho em tarefas de depuração. Esta significância é grande o suficiente para que programadores, mesmo tendo mais tempo de experiência, apresentem desempenho consistentemente inferior em tarefas de depuração. Este aspecto não é explorado em trabalhos como os de Uwano et al. (2006) e Bendarik et al. (2006), enquanto existem até mesmo abordagens que eliminam este aspecto completamente, certificando-se de que os voluntários não conheçam a linguagem de programação

utilizada e incluindo o treinamento na linguagem como parte do experimento (KO; UTTL, 2003).

Destaca-se também a análise estatística das transições entre os blocos semânticos do programa. Nos trabalhos relacionados encontrados, não foi observada este tipo de análise. A partir dos experimentos realizados, foi possível constatar que os participantes passaram a maior parte do tempo de leitura no bloco de processamento e que as transições entre os blocos são muito frequentes e dinâmicas.

Por fim, a compreensão dos processos cognitivos envolvidos na leitura de programas de computadores é uma tarefa complexa e que exige uma investigação mais profunda. Como trabalhos futuros, sugere-se incluir o relato dos voluntários ao final da leitura do código, para adquirir mais informações sobre como o aluno conclui onde está o erro. Recomenda-se também a realização do experimento com profissionais experientes em programação, para verificar se eles empregam estratégias mais eficientes. Acredita-se que a compreensão das estratégias de leitura de códigos de computador pode beneficiar o ensino dos conteúdos relacionados à programação, na medida em que as estratégias mais eficientes possam ser ensinadas para os alunos iniciantes.

EYE TRACKING-BASED ANALYSIS OF CODE READING STRATEGIES EMPLOYED BY STUDENTS

ABSTRACT

This paper presents an experiment to analyze the strategies used during program reading in C language. We used an eye tracker to determine the gazed regions in the code and which reading strategies were used in these regions. To determine the gazed areas, we captured the fixations in each line of the program. The task performed by the students was to find a logical bug in four different codes. The eye movement analysis has shown the occurrence of the first scan pattern described in literature. Division of the code in semantic blocks revealed that the students spend most time in the processing block and that the transitions between blocks occur in a very dynamic way. The results indicate that variations in the syntax of the programming languages used are an aspect as important as programming experience when debugging simple logical errors in computer program source-code.

KEYWORDS: Computer program comprehension. Eye tracker. Fixations.

AGRADECIMENTOS

Agradecimentos à Fundação Araucária, Secretaria de Estado da Ciência, Tecnologia e Ensino Superior (SETI-PR) e ao Governo do Estado do Paraná, pelo apoio financeiro recebido.

REFERÊNCIAS

BEDNARIK, R**. Methods to analyze visual attention strategies: Applications in the studies of programming.** Tese de doutorado. University of Joensuu, 2007.

BEDNARIK, R. et al. Program visualization: Comparing eye-tracking patterns with comprehension summaries and performance. In: Proceedings of the 18th Annual Psychology of Programming Workshop. University of Sussex. 2006. p. 66-82.

BUSJAHN, T. et al. Eye tracking in computing education. In: Proceedings of the tenth annual conference on International computing education research. ACM, 2014. p. 3-10.

CROSBY, M.; STELOVSKY, J. How do we read algorithms? A case study. **Computer**, v. 23, n. 1, p. 25-35, 1990.

DU BOULAY, B.; O'SHEA, T.; MONK, J. The black box inside the glass box: presenting computing concepts to novices**. International Journal of Man-Machine Studies**, v. 14, n. 3, p. 237-249, 1981.

DUCHOWSKI, A. **Eye tracking methodology: Theory and practice**. Springer Science & Business Media, 2007.

FUCHS, A. F. The saccadic system. In: BACH-Y-RITA, P. (org). **The control of eye movements**. Elsevier, 2012. p. 343-362.

GOLDBERG, J. H.; WICHANSKY, A. M. Eye tracking in usability evaluation: A practitioner's guide. In: HYONA, J; RADACH, R; DEUBEL, H. (org). **The mind's eye: Cognitive and applied aspects of eye movement research**. 2003. p. 493-516.

JUST, M. A.; CARPENTER, P. A. A theory of reading: from eye fixations to comprehension. **Psychological review**, v. 87, n. 4, p. 329-354, 1980.

KO, A. J.; UTTL, B. Individual differences in program comprehension strategies in unfamiliar programming systems. In: 11th IEEE International Workshop on Program Comprehension, 2003, p. 175-184.

MALONEY, J. et al. Scratch: programming for all. **Communications of the ACM**, v. 52, n. 11, p. 60-67, 2009.

RAYNER, K. Eye movements in reading and information processing: 20 years of research. **Psychological bulletin**, v. 124, n. 3, p. 329-354, 1998.

SHARIF, B.; FALCONE, M.; MALETIC, J. I. An eye-tracking study on the role of scan time in finding source code defects. In: Proceedings of the Symposium on Eye Tracking Research and Applications. ACM, 2012. p. 381-384.

SMI. **RED 250 / RED 500**. Disponível em < http://www.smivision.com/en/gaze-and-eye-tracking-systems/products/red250-red-500.html >. Acesso em julho de 2016a.

SMI. **SMI Experiment Suite 360°**. Disponível em < http://www.smivision.com/en/gaze-and-eye-tracking-systems/products/experiment-suite-360-software.html >. Acesso em julho de 2016b.

TAN, P.; TING, C.; LING, S. Learning difficulties in programming courses: undergraduates' perspective and perception. In: International Conference on Computer Technology and Development. Kota Kinabalu, Malaysia, 2009. p. 42-46.

UWANO, H. et al. Analyzing individual performance of source code review using reviewers' eye movement. In: Proceedings of the 2006 symposium on Eye tracking research & applications. ACM, 2006. p. 133-140.

ZÜLCH, G.; STOWASSER, S. Eye tracking for evaluating industrial human–computer interfaces. In: HYONA, J; RADACH, R; DEUBEL, H. (org). **The mind's eye: Cognitive and applied aspects of eye movement research**. Elsevier, 2003. p. 531-553.

DOI: 10.3895/rbect.v10n1.5688
Como citar: ALBERTON, B. V.; LEE, D. G.; LIE, M. M. I.; GAMBA, H. R.; BORBA, G. B. Utilização de um rastreador ocular para análise de estratégias de leitura de programas de computador por discentes. **Revista Brasileira de Ensino de Ciência e Tecnologia**, v. 10, n. 1, 2017. Disponível em: <https://revistas.utfpr.edu.br/rbect/article/view/5688>. Acesso em: xxx.
.