Avaliação da Legibilidade de Programas Escritos por Alunos Iniciantes

Eliane Cristina de Araujo[1], Dalton Serey Guerrero[1], Jorge César Abrantes de Figueiredo[1]

[1]Departamento de Sistemas e Computação
Universidade Federal de Campina Grande – Campina Grande, PB – Brasil

{eliane,dalton,abrantes}@computacao.ufcg.edu.br

Abstract. *In this paper, we report on a study that was carried out, in the context of an introductory programming course, to investigate how code readability correlates with the students' achievements. We suggest a simple metric to automatically assess beginners code readability. The study revealed a correlation between code readability and students' course performance. In parallel, we brought to light other factors which, taken together with the readability metric, can better explain students performance in the course.*

Resumo*. Neste artigo, apresentamos um estudo realizado, no contexto de um curso de introdução à programação, em que investigamos a relação entre a legibilidade dos programas produzidos pelos alunos e o seu desempenho na disciplina. Propusemos e avaliamos uma métrica simples de legibilidade de código Python para programas produzidos por estudantes de programação introdutória. Nossos resultados confirmam que há uma correlação entre a métrica de legibilidade dos programas e o desempenho dos alunos, indicando que a métrica captura um aspecto considerado pelos professores na avaliação dos programas.*

1. Introdução

O desenvolvimento de programas por estudantes em cursos introdutórios de programação é uma das atividades que mais pode contribuir na aprendizagem desta disciplina. Entretanto a avaliação destes programas e o consequente feedback dado ao aluno, demanda muito trabalho do professor. Por essa razão, professores têm optado por diminuir a quantidade de programas/exercícios propostos para os alunos (Cheang et al 2003).

Os sistemas de verificação e testes automáticos de programas podem reduzir a carga dos professores e viabilizar o estímulo à produção de muitos programas pelos estudantes. Nesse modelo, os programas são verificados automaticamente e se produz um feedback imediato para o estudantes sobre o programa-solução produzido (Campos et al, 2004). Os verificadores, por se nortearem pelos testes automáticos, consideram para avaliação apenas a resposta produzida pelos programas para um conjunto pré-definido de casos de teste, desconsiderando as qualidades internas do código submetido pelo estudante.

Neste trabalho, apresentamos os resultados de um estudo cujo objetivo foi analisar como a legibilidade dos códigos dos alunos pode ser avaliada de forma automática e como ela se relaciona com o desempenho dos estudantes no curso. Para isso, propusemos uma métrica para avaliar a legibilidade do código produzido por

programadores iniciantes e que pode ser obtida de forma automática. Neste estudo procuramos verificar em que medida a capacidade de escrever programas mais legíveis pode estar correlacionada com o desempenho do aluno no curso.

O trabalho está organizado da seguinte forma: a Seção 2 apresenta o referencial teórico e alguns trabalhos relacionados à legibilidade de códigos. A seção 3 explica a metodologia adotada para realizar o estudo e descreve o contexto de produção dos dados relatando a forma como foram coletados e tratados. A Seção 4 apresenta os resultados alcançados pelo estudo, que são analisados e discutidos na seção 5. Os comentários finais, bem como as idéias para refinar e melhorar o trabalho, são tratados na seção 6.

2. Referencial Teórico e Trabalhos Relacionados

Estudos apresentam diferentes definições sobre a noção de legibilidade. Posnett et al. (2011) consideram que a legibilidade é a impressão subjetiva que os programadores têm sobre o quão difícil de entender é determinado código. Para estes autores, um trecho de código é dito legível se for fácil de ler, compreender e manter. No entanto, esta noção de legibilidade está fortemente relacionada a fatores humanos cognitivos, o que a torna difícil de quantificar e medir. Buse et al. (2010) propuseram uma métrica para medir a legibilidade do código com base em uma pesquisa experimental envolvendo anotadores humanos. Posnett apresentou uma outra proposta, seguindo a mesma linha.

Os modelos para legibilidade de código tanto de Buse quanto de Posnett, foram obtidos experimentalmente observando códigos produzidos sob condições diferentes dos programas de alunos iniciantes. Em geral, os programas de iniciantes são respostas para exercícios de programação propostos pelos professores, podendo ser abertos ou não (Blinkstein, 2011). Estes programas são compostos geralmente por poucas linhas de código. Além disso, cursos introdutórios de programação tendem a não optar pelo paradigma de orientação a objetos, mesmo utilizando linguagens que dêem este suporte. Os programas avaliados pelos autores citados na obtenção de suas métricas são escritos em linguagens orientadas a objetos. Só este fator já é motivo de impedimento para a utilização das métricas por eles propostas.

Neste trabalho, propusemos uma métrica para avaliar a legibilidade dos programas dos estudantes considerando a adesão ao padrão de codificação estabelecido pela comunidade Python que é a linguagem de programação adotada para a escrita de programas no curso. Esta simplificação da análise da legibilidade é amparada pela experiência da comunidade de prática da linguagem através dos objetivos do Guia de Estilo para Código Python, conhecido como PEP08 (Python, 2013) "(...) as regras aqui descritas objetivam melhorar a legibilidade dos códigos Python e torná-los consistentes com o amplo espectro de códigos Python desenvolvidos mundialmente.".

Com uma ferramenta de verificação de conformidade ao padrão PEP08, medimos a quantidade de itens do código que estão em desacordo com este guia de estilo. Neste estudo, a métrica legibilidade foi reduzida em sua complexidade e quantificada de forma inversa: medindo o número de PEP08-defeitos (ao que chamaremos apenas de *defeitos*). Por exemplo: o programa 1 será mais legível que o programa 2 se ele apresentar menos *defeitos* de legibilidade que o outro.

3. Metodologia

A pesquisa foi realizada no contexto da disciplina Programação I do curso de Bacharelado em Ciência da Computação da UFCG. É dado um grande enfoque à resolução de problemas e ao desenvolvimento de muitos programas. Essa abordagem é suportada por um ferramental técnico de apoio ao ensino desenvolvido pela própria equipe pedagógica. Em síntese, do ponto de vista do estudante, o ferramental permite que ele: tenha acesso ao enunciado dos exercícios propostos cuja resposta é um pequeno programa ou função e envie sua resposta ao professor. Estes sistemas ampliaram a possibilidade de interação com o aluno fora de sala de aula através da Web. Desta forma foi possível, atendendo às necessidades da disciplina, aumentar a quantidade de exercícios propostos sem prejuízo para o feedback dado ao aluno. O sistema viabiliza a correção mais rápida e padronizada dos exercícios.

Neste contexto, realizamos um estudo de caso utilizando dados do curso ministrado no período de 2011.2. Para a realização do estudo verificamos os programas dos alunos coletados após submissão ao sistema de correção e armazenados no banco de respostas. A submissão do programa indica, em sua maioria, que o aluno considera que aquela é uma implementação válida para especificação do programa dada no exercício. Assumimos que cada exercício proposto ao aluno é a especificação de um programa. Os exercícios que são propostos ao longo do curso são corrigidos automaticamente através do testador automático. Os exercícios propostos nas provas são testados automaticamente e também corrigidos por um professor, ou seja, há uma avaliador humano para checar além da corretude outros atributos como: desempenho, eficiência, estilo, complexidade, legibilidade, etc.

3.1. Medidas, variáveis e conjunto de dados

A questão de pesquisa que norteou o estudo foi: Como a qualidade – corretude e legibilidade – e a quantidade de código produzido pelo estudante ao longo do curso podem influenciar seu desempenho?

A *corretude* dos programas foi medida em função da nota emitida pelo testador automático, que é uma ponderação entre os casos de testes em que o programa é bem sucedido, a relevância e a quantidade de casos de testes totais formuladas pelo professor. A *quantidade* de código produzida pelo aluno no período de estudo foi medida em termos de questões para as quais foi submetida pelo menos uma solução e também em número de linhas de código destes programas (LoC, tradicional métrica da engenharia de software). O *desempenho* do estudante no curso foi medido em função de sua nota na segunda prova da disciplina. Optou-se por analisar as notas da segunda prova por ser uma avaliação central no curso, cobrindo aproximadamente 70% do conteúdo. A *legibilidade* dos programas foi medida considerando o padrão de codificação estabelecido pelo PEP08. Como já ressaltado, medimos o número de não-conformidades ao padrão encontradas no código, ou seja o número de *defeitos*. A *densidade* de defeitos é a medida que procura quantificar a habilidade que o aluno tem em escrever programas com menos defeitos, ou seja mais legíveis. Para tanto, medimos a quantidade de *defeitos* de um conjunto de programas e dividimos pelo seu número de *linhas de código* (LoC).

Os dados que utilizamos no estudo referem-se aos programas desenvolvidos pelos estudantes e as notas atribuídas pelo testador automático, com base nos testes propostos

pelos professores. Foram coletados 170 programas de 76 alunos. Consideramos os programas submetidos pelos estudantes no período compreendido entre duas provas da disciplina (intervalo de aproximadamente de 30 dias). Apenas a última versão submetida de cada questão enviada ao sistema por cada aluno, foi considerada para a análise. Os dados coletados neste período correspondem a 64 exercícios diferentes. No total, para esta fase, coletamos 3080 programas de 76 alunos.

4. Resultados

Observamos uma série de programas escritos pelo aluno e calculamos o índice *densidade defeitos*. Procuramos estabelecer a correlação entre este valor e a nota do aluno na segunda prova da disciplina, mostrado na Tabela 1. Além disso, aprofundamos a análise dos mesmos fatores através do agrupamento dos indivíduos de acordo com a nota. Usamos o método de Spearman para o cálculo da correlação, já que não verificamos normalidade no conjunto de dados.

	Questões	LoC	Defeitos	Densidade
Nota do aluno	0,550	0,530	0,365	0,056

Tabela 1 – Valores dos coeficientes de correlação

A Tabela 1 mostra os coeficientes de correlação entre a nota da prova e as variáveis: número de exercícios resolvidos, número de linhas de códigos produzidas - LoC, número de defeitos de legibilidade e densidade de defeitos de legibilidade. Há uma correlação moderada/forte (0,550) entre o número de questões resolvidas por cada aluno com sua nota na segunda prova. A mesma força, também é observada na correlação entre o número de linhas de código produzidas pelo aluno e a nota da prova (0,530). Chama à atenção o valor positivo encontrado para o coeficiente de correlação entre a nota na prova e a densidade de defeitos de legibilidade (0,056), embora em valor absoluto, seja inexpressivo. Este valor parece contrariar resultados obtidos em um estudo correlacional anterior quando avaliamos os programas individualmente, mostrando que para programas corretos, quanto menos legível o programa menor será sua nota por um avaliador humano. Tal diferença nos motivou a refinar o estudo, agrupando estudantes em função de seu desempenho geral na disciplina, cujo resultado pode ser visto na Figura 1.

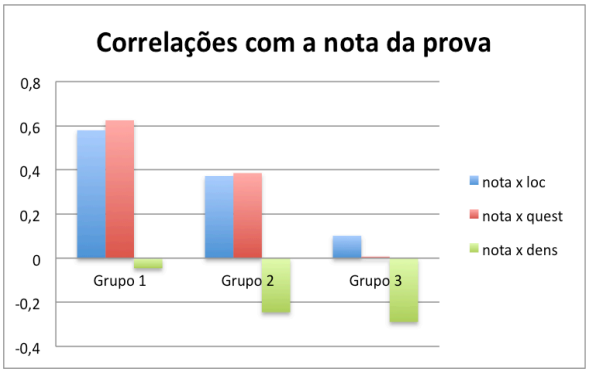

Figura 1 - Coeficientes de correlação entre a nota atribuída pelo professor e os fatores número de linhas de código produzidas, número de questões resolvidas e densidade de defeitos de legibilidade em cada grupo de estudantes. Grupo 1 = regular ou fraco. Grupo 2 = bom. Grupo 3 = excelente.

Os alunos foram agrupados em três grupos. No grupo 1 estão os estudantes com notas inferiores a 7.5, com desempenho considerado de regular a fraco. No grupo 2, estão os estudantes com notas entre 7.5 e 8.7, com desempenho considerado bom. E, finalmente, no grupo 3, os estudantes com notas acima de 8.7, com desempenho considerado excelente pelos professores. Novamente produzimos os coeficientes de correlação entre as notas na segunda prova e os fatores LoC, número de questões e densidade de defeitos de legibilidade.

O gráfico da Figura 1 mostra que para o Grupo 1, os alunos com desempenho regular ou fraco, o volume de programação realizado até a data da prova está mais correlacionado com nota do aluno do que a densidade de defeitos de legibilidade. Em contraste, para os alunos do Grupo 3, o valor absoluto da densidade de defeitos de legibilidade é bem superior aos outros fatores neste grupo. Isto indica que na análise das notas dos alunos do Grupo 3, as maiores notas são dos alunos com menor densidade de defeitos de legibilidade em seus códigos. Os resultados encontrados para o Grupo 2, mostram uma correlação negativa moderada entre densidade de defeitos e a nota dos alunos. Contudo, o valor absoluto da força desta correlação é menor que a dos outros dois fatores, consideradas correlações moderadas.

5. Análise e Discussão

A correlação moderada entre a quantidade de código produzida e a nota do aluno na prova corrobora com a noção intuitiva de que quanto mais o aluno pratica, produzindo código, melhor será a sua nota. Entretanto, a análise da qualidade do código, no quesito legibilidade deve ser ponderada mais cuidadosamente. A correlação observada entre o total de defeitos de legibilidade acumulados nos programas do aluno até a data da prova e nota nesta prova é de 0,365. O que parece indicar que, mesmo de forma moderada, quanto maior a quantidade de defeitos maior a nota do aluno. Temos que considerar, contudo, que a quantidade de defeitos é função das variáveis número de questões resolvidas e total de linhas de código produzidas. Uma forma melhor de interpretar essa correlação seria: quanto mais o aluno pratica e, portanto, quanto mais se dispõe a errar, mais chances tem de ter um bom desempenho.

A medida de densidade de defeitos procura eliminar a influência dos fatores ligados ao volume de prática de cada estudante, dividindo a quantidade de defeitos pelo número de linhas de código do aluno. Neste caso, contudo, a correlação observada foi praticamente inexistente (0,056). Agrupamos os estudantes de acordo com sua nota, a fim de obter resultados mais esclarecedores. Observa-se que a correlação entre a nota do aluno e a densidade de defeitos de legibilidade dos seus programas aumenta à medida que aumenta a nota do estudante (em valores absolutas, temos para o grupo 1: 0,05; para o grupo 2: 0,27; e para o grupo 3: 0,33). Esse fenômeno parece indicar que há uma lógica na forma de avaliação dos programas que é dominada pela componente corretude. Isto é, o professor só parece levar em conta a questão da legibilidade dos programas, depois que o programa é considerado minimamente correto, em termos do número de casos de teste a que satisfaz. Ainda assim, é necessário observar que ou a densidade de defeitos de legibilidade não é o único fator envolvido ou a métrica não captura adequadamente o conceito de legibilidade adotado pelo professor.

6. Conclusões e Trabalhos Futuros

Neste trabalho realizamos um estudo para investigar como a legibilidade dos códigos dos alunos relaciona-se com o desempenho dos estudantes no curso. Para isso, propusemos uma métrica simples e automática para o cálculo da legibilidade dos códigos produzidos por programadores iniciantes.

Como resultado deste trabalho, verificamos que o desempenho do estudante é fortemente correlacionado com a quantidade de código por ele produzida ao longo do curso. No geral, o estudo mostra que produzir programas legíveis é menos relevante para o desempenho do que produzir muitos programas. Na análise por grupos de acordo com o desempenho, no entanto, a legibilidade dos códigos é um fator relevante quanto melhor for o desempenho do aluno.

O estudo mostra que, além da corretude, a legibilidade é um fator analisado pelo professor na composição da nota. É importante questionar, como um caminho natural para uma futura pesquisa, quais são os outros fatores? Também interessaria repetir o estudo com outros conjuntos de dados, a fim de dar maior sustentação aos resultados.

Incluir legibilidade de código e outros aspectos de qualidade interna em uma suíte de testes automáticos enriquece o feedback dado aos estudantes. Isto pode ser útil não só para os cursos regulares/presenciais de programação que são fortemente baseados em resolução de problemas como também nos MOOCs – Massive Open Online Courses onde a escala é uma questão relevante.

Referências

Blinkstein, P.. 2011. “Using learning analytics to assess students' behavior in open-ended programming tasks”. In: Anais do 1st International Conference on Learning Analytics and Knowledge (LAK '11). ACM, New York, NY, USA, 110-116.

Buse, R. P. L. e Weimer, W. R.. 2010. “Learning a Metric for Code Readability”. In: IEEE Trans. Softw. Eng. 36, 4 (Julho 2010), 546-558.

Campos, C. P. e Ferreira, C. E. . “BOCA: um sistema de apoio a competições de programação (BOCA: A Support System for Programming Contests)”. In: Workshop de Educação em Computação (Brazilian Workshop on Education in Computing), 2004, Salvador. Anais do Congresso da SBC, 2004.

Cheang B., Kurnia A., Lim A., e Oon W.. 2003. “On automated grading of programming assignments in an academic institution”. In: Comput. Educ. 41, 2 (setembro 2003), 121-131. 30-7

Posnett, D., Hindle, A., e Devanbu, P. “A simpler model of software readability”. In: Anais do 8th Working Conference on Mining Software Repositories (MSR '11). ACM, New York 2011, NY, USA, 73-82.

Style Guide for Python Code. http://www.python.org/dev/peps/pep-0008/#introduction. [Online. Acesso 01-março-2013].