UMA ABORDAGEM PARA OTIMIZAÇÃO DA QUALIDADE DE CÓDIGO FONTE BASEADO NA COMPLEXIDADE ESTRUTURAL

Heleno de S. Campos Junior[1], Luís Rogério V. Martins Filho[2], Marco Antônio P. Araújo[3]

1. Estudante de Bacharelado em Sistemas de Informação do IF Sudeste MG – Campus de Juiz de Fora;
2. Estudante de Bacharelado em Sistemas de Informação do IF Sudeste MG – Campus de Juiz de Fora; *luisrogeriojf@gmail.com
3. Professor do Núcleo de Informática do IF Sudeste MG – Campus de Juiz de Fora.

Palavras Chave: *Complexidade Ciclomática, Complexidade Desnecessária, Engenharia de Software.*

Introdução

Com o aumento do uso de software nos mais variados contextos, cresce a exigência por produtos mais abrangentes, que acabam sendo desenvolvidos sem a preocupação com a qualidade estrutural do código fonte, devido à demanda por entrega cada vez mais rápida. Uma métrica que é estudada para medir a qualidade do software é a complexidade ciclomática, que entre outras aplicações, mede a sua dificuldade de entendimento e de manutenção [1]. É definida a partir do Grafo de Fluxo de Controle [2], que ilustra através de um grafo o fluxo das estruturas de comando do algoritmo. Verificado isso, nota-se que existe a necessidade de estudos no apoio ao desenvolvimento, manutenção e evolução de software, que levem em conta seus aspectos estruturais.

Resultados e Discussão

Para averiguar as origens do problema estudado, realizou-se um estudo exploratório tendo como objeto de pesquisa avaliações da disciplina de Algoritmos de alunos de graduação. De 482 códigos fonte analisados, evidenciou-se um problema que chamamos de complexidade estrutural desnecessária em 16%, que ocorre quando algoritmos são desenvolvidos com uma complexidade ciclomática que pode ser diminuída, sem alterar o comportamento do código fonte. A partir desse problema, desenvolveu-se uma ferramenta para identificar a possibilidade de diminuição da complexidade ciclomática de um algoritmo e ainda exibição de um grafo de fluxo de controle representando o código fonte otimizado, como mostram as Figuras 1 e 2, respectivamente. A partir da análise desse grafo, o programador pode refatorar o código fonte levando em conta as melhorias observadas.

A abordagem desenvolvida foi avaliada através de um estudo de desempenho, cujos valores obtidos são descritos na tabela.

Tabela 1. Indicadores de desempenho da abordagem.

Indicador	Descrição	Valor
Acerto	Porcentagem total de acertos	100%
Cobertura	Proporção entre casos detectados e casos otimizáveis	74,32%
Precisão	Proporção entre casos detectados e total de casos analisados.	95,75%

Além da avaliação de desempenho, foram planejados e executados dois estudos experimentais, de acordo com as diretrizes propostas em [3]. A população desses estudos foi composta de 19 alunos de Engenharia Mecatrônica na disciplina de Algoritmos e 10 alunos de Sistemas de Informação na disciplina de Linguagem de Programação II. Após análise estatística dos dados coletados, evidenciou-se que o uso da ferramenta auxiliou no desenvolvimento de código fonte com uma complexidade ciclomática menor do que quando do não uso.

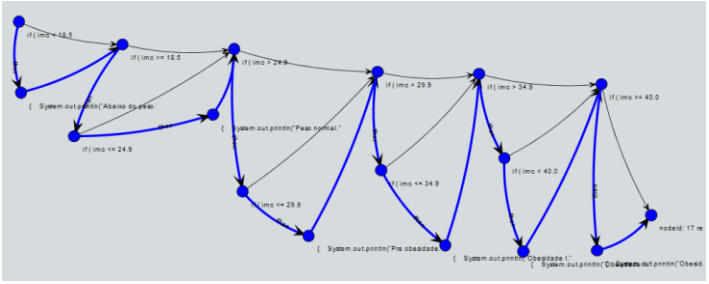

Figura 1. Exemplo de Grafo de Fluxo de Controle com complexidade desnecessária.

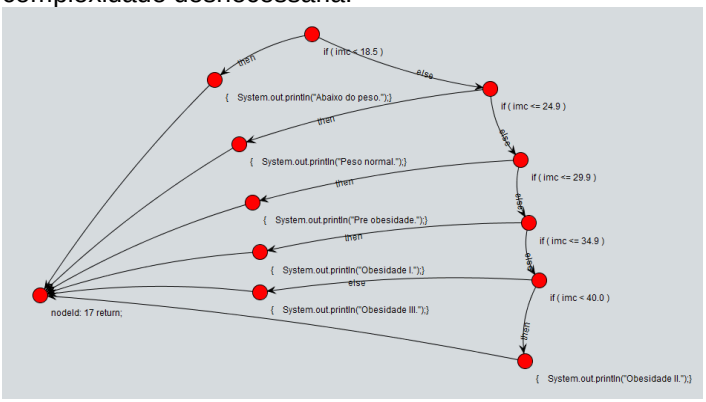

Figura 2. Grafo de Fluxo de Controle otimizado exibido pela ferramenta.

Conclusões

Neste trabalho, através das evidências encontradas nos estudos, é possível concluir que um dos contextos onde a complexidade estrutural desnecessária acontece é entre aprendizes de programação e que a ferramenta desenvolvida auxilia seus usuários a desenvolver código fonte com complexidade ciclomática otimizada nesse contexto, gerando maior manutenibilidade e legibilidade do código fonte.

Pretende-se ainda avaliar a ferramenta em outros contextos, evoluir a abordagem para abranger mais casos de complexidade desnecessária e realizar a refatoração do código fonte analisado. Ainda, estudar os impactos na manutenção, evolução e desenvolvimento de software.

Agradecimentos

Agradecimentos ao IF Sudeste MG por apoiar a pesquisa através do fomento de bolsas aos autores.

[1] McCABE, T. J. A complexity measure. IEEE Trans. on Software Eng. Vol. SE-2, N. 4. p. 308-320, 1976.

[2] ALLEN, F. E. Control flow analysis. Symposium on Compiler optimization, Proceedings of a Symposium on Compiler optimization, Urbana-Champaign, Illinois , p.1-19, 1970.

[3] WOHLIN, C.; RUNESON, P.; HÖST, M., OHLSSON, M. C.; REGNELL, B., WESSLÉN, A.; Experimentation in Software Engineering. Springer Science & Business Media, 2012.