アイトラッキングを用いた単純なソースコードの定量的品質評価
－インデントが読みやすさに与える影響の分析－

Quantitative Quality Assessment of Simple Source Code Using Eye Tracking
- Analysis of the Impact of Indentation on Readability -

枡野 真道[*1]，滝野 了太[*1]，頼本 康[*1]，前田 暉正[*2]，松本 慎平[*1]
Masamichi MASUNO[*1], Ryouta TAKINO[*1]
Ko YORIMOTO[*1], Terumasa MAETA[*2], Shimpei MATSUMOTO[*1]
[*1] 広島工業大学情報学部
[*1] Faculty of Applied Information Science，Hiroshima Institute of Technology
Email: {bm21121, bm21075, bm20141, s.matsumoto.gk}@cc.it-hiroshima.ac.jp
[*2] 広島工業大学大学院工学系研究科
[*2] Graduate School of Science and Technology，Hiroshima Institute of Technology
Email: md22006@cc.it-hiroshima.ac.jp

あらまし：本研究では，インデントの設定に焦点を当て，アイトラッキングを用いてプログラムの読解過程を解析することにより，インデントが読みやすさに与える影響を明らかにする．ソフトウェアライフサイクルにおける作業量の中で，保守作業が占める割合は非常に高いと言われている．特に，保守の全行程の中でも，ソースコードの内容理解，すなわち読解が最も時間的コストが高い作業となっている．このような背景から，ソースコード読解力は重要なプログラミング能力のひとつと位置付けられている．読解と共に，ソースコードの可読性について意識を向けさせることも，重要な読解学習のひとつとされている．ソースコードの読みやすさを左右する要因としては，コードの構造，命名規則，コメントの有無や品質，インデントや括弧の位置などが挙げられる．しかし，これらの要因が読みやすさに与える影響を定量的に示した研究については，十分に行われていない．そこで本研究では，インデントが読みやすさに与える影響に着目し，アイトラッキングを用いてプログラムの読解過程を分析した．その結果，小規模なソースコードの場合，インデントがない場合でもスライシングに負の影響を与えない可能性が示唆された．
キーワード：アイトラッキング，ソースコード，インデント，読みやすさ

1. はじめに

ソフトウェアライフサイクルにおける作業量の中で，保守作業が占める割合は非常に高いと言われている(1)．実際，ソフトウェアの開発から運用，そしてその終焉に至るまでの一連のプロセスを通じて，保守作業は最もコストと時間がかかる工程であると言われている(2)．特に，保守の全行程の中でも，ソースコードの内容理解，すなわち読解が最も時間的コストが高い作業となっている．このような背景から，ソースコード読解力は重要なプログラミング能力のひとつと位置付けられている．読解力の向上は，バグの修正，ソフトウェアの性能や安定性の維持だけでなく，システムの改善や新たな機能の追加といった開発作業の効率化にも直結する．さらに，ソースコード読解力は，他者とのコードレビュー時にも重要な役割を果たす．よって，ソースコードの読解力を高める教育は，高等教育機関などで従来から実践されてきた．

近年の研究では，ソースコード読解の学習とその効果について様々な観点から調査されている．例えば，注釈付きのソースコードの提供や，ソースコードの構造を視覚的に示すことが読解力の向上に寄与することが報告されている．ソースコードの構造を可視化するオープンソースのツールも公開されており，教育現場だけではなく，開発現場でも実際に広く利用されている．

ソースコードの可読性について意識を向けさせることも，重要な読解学習のひとつである．ソースコードの読みやすさを左右する要因としては，コードの構造，命名規則，コメントの有無や品質，インデントや括弧の位置などが挙げられる．しかし，これらの要因が読みやすさに与える影響を定量的に示した研究については，十分に行われていない．

そこで本研究では，インデントの設定に焦点を当て，アイトラッキングを用いてプログラムの読解過程を解析することにより，インデントが読みやすさに与える影響を明らかにする．

2. 提案

本研究では，ソースコード教材の重要な要素であるインデントに着目した．そして，インデントの設定が読解精度への影響の原因をアイトラッキングから明らかにすることを目的とする．本研究では，スライシングの精度を読解精度と定義する．あるソースコードを処理した後の変数の値を求めるタスクを与えたとき，そのタスクを解決するために必要な行は，バックワードスライシングにより明確である．この特性を利用すれば，タスクを解くために必要の

無い行を見ていた度合いを定量化すれば，読解の効率を算出できる．本研究では，この読解効率を読解精度と呼ぶ．なお，タスクを解くために必要の無い行を，本研究では「ダミー行」と呼ぶ．

不要な行に対する注視の程度については，読解教材の正解率や，学習者に対するインタビューを行うことである程度評価することができる．しかし，学習者の評価の理由や，正解率を高く/低くした理由を明らかにすることは容易ではない．この点に関して，アイトラッキングが有用である．アイトラッキングを行うことで，学習者がソースコードのどの部分を見て混乱しているか，あるいは理解に時間がかかっているかを知ることができる．

アイトラッキングから得られたデータを分析する手法として，単純マルコルモデルによる各ノードへの遷移確率を用いる．単純マルコフモデルとは，不規則に変化するシステムをモデル化するために使用される確率モデルで，将来の状態は現在の状態のみに依存し，それ以前に発生したイベントには依存しないと想定する．仮説として，コード内にダミー行を設けることで適切なインデントが用いられている教材ほどダミー行への遷移確率が低くなると考えた．

本研究では，読解教材の重要な要素となるインデントの違いによる影響調査する．インデントの設定については，一般的に用いられている 4 文字分の幅と一切インデントが行われていないもの，インデントを不規則に配置したものの 3 つを比較する．なお，ソースコードの内容は変えず，提示方法のみを変化させる．その際，出題する順序を被験者毎に不規則にすることで，課題の提示順の影響をできる限り排除する．

3. 実験結果

本研究では，C 言語の基本を理解している大学生及び大学院生 17 名を被験者とする． まず，被験者の事前の知識をプレテストにより把握する．次に，アイトラッキングを行いながら，読解タスクを被験者に解いてもらう．タスクは全部で 9 種類用意する．内訳は，インデントが 4 文字幅もの，ないもの，不規則なもの，それぞれ 3 問ずつとする．それぞれのコード内容は同じものであるため，慣れの影響をできる限り回避するため，出題する順序を被験者ごとに変化させる．今回，制限時間は設定しなかった．実験後，主観評価を得た．主観評価は認知負荷の質問[3]に従う．実験後，アイトラッキングのデータを分析し，得られた結果の理由を定量的に明らかにする．

被験者には，プログラミング能力を判定するテストを実施し，成績順に A，B，C に群分けした．人数構成については，A は 7 名，B は 5 名，C は 5 名であった．それぞれのインデント設定でのダミー行への遷移確率を図 1 に示す．インデントを不規則に設定した場合，プログラミング能力とダミー行への遷移確率に負の関係が見られた．そして，Welch の t 検定の結果，A 群と C 群の間に統計的に有意($p<.05$, 両側)な差が確認された．この結果から，インデントを使った「読みやすさ」という曖昧な表現について，「プログラミングが得意でない人のスライシングの精度を高めるための配慮」というより厳密な説明が可能であることが示唆された．なお，インデント「無し」の場合にプログラミング能力による差が見られなかった点も注目すべき点である．今回のタスクは，基本的かつ小規模なコードであった．この結果は，小規模・単純な制御構造のプログラムの場合，インデントの設定は読解効率に大きな影響がないため，学習者は不適切なインデントの設定でソースコードを書きがちであるという可能性を示唆している．

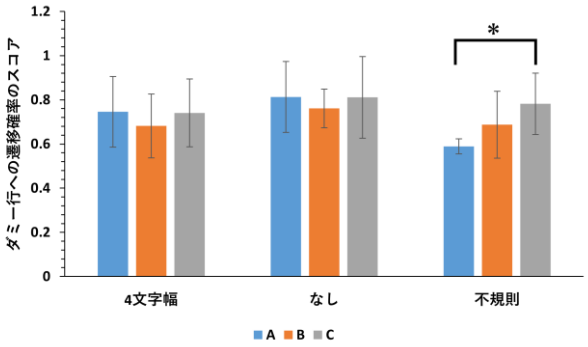

図 1 インデントの状態による成績ごとのダミー遷移確率

実験後に行なった主観評価については，Welch の t 検定の結果，課題外在性負荷，学習関連負荷において，インデント設定の違いに応じて有意な差($p<.05$ 両側)が確認された．この結果は，被験者の主観においては，インデントの影響は見られることを示唆している．また，学習関連負荷においても有意な差($p<.05$)が見られた．以上から，相互レビューなどを学習に含める場合，インデントの設定について，教授者は特に強調する必要性が示唆された．

4. おわりに

本研究では，インデントがソースコードの読みやすさに与える影響を調査した．その結果，小規模なソースコードの場合，インデントがない場合でもスライシングに影響を与えない可能性が示唆された．

謝辞

本研究は，独立行政法人日本学術振興会科学研究費助成事業(基盤研究(C)20K0319，22K02815)の助成を受けて実施した成果の一部である．

参考文献

(1) Boehm, B., and Basili, V. R. (2001). Defect reduction top 10 list. Computer, 34(1), 135-137.
(2) Goldberg, A. (1987). Programmer as reader. IEEE Software, 4(5), 62.
(3) J. Sweller, Element Interactivity and Intrinsic, Extraneous, and Germane Cognitive Load, Educational Psychology Review, 22, pp.123-138 (2010).