B-02

開発経験によるソースコード読解時間の影響分析

An Analysis of Impact of Development Experience on Source Code Comprehension Time

吉岡 俊輔†　森崎修司†　松本健一†
Shunsuke Yoshioka　Shuji Morisaki　Ken-ichi Matsumoto

1. 研究背景

情報システムは社会の至るところに普及しており，その依存度は年々高まっている．また，社会からの情報システムに対する機能や品質の要求も年々高まっており，ソフトウェアの開発規模は増加傾向にある．加えて，システムの消費サイクルも昔に比べ短くなっている．そのため，短期間で高品質・高機能な情報システムを開発する必要性が高まっており，保守開発・派生開発の増加が目立つ[1]．その結果，品質管理がソフトウェア開発において大きな問題となっている[2].

保守開発や派生開発では，既存のソフトウェアに改変を施し機能追加を行う．このような開発では，既存ソフトウェアの内容を理解して，変更，拡張する必要があり，非常に小さな改変であっても，時間がかかる場合が多い．本稿はソースコード理解を対象として，開発経験がソースコードの理解時間に与える影響を分析する．また開発経験の長い開発者とそうでない開発者の間に，ソースコードの読解技術にどのような差があるか分析し，どのように読解の仕事を割り当てるのが合理的か示す.

2．実験

2.1　概要

本実験では，既存のソースコードに変更を加えた場合，その変更によって問題が発生するかどうかを開発者が判断するのに，どれだけの時間がかかるかを計測し，開発者の経験が読解時間にどのように影響を与えるかを明らかにすることを目的としている．

実験をするに当たり以下のものを用いた．

116 人の被験者，被験者へのアンケート，実験用ソースコード（以下，ソースコード（version1））, ソースコード（version1）に変更を加えた計 13 種類の新たなソースコード（以下，ソースコード（version2））.

「被験者」はソフトウェア開発の実務経験者から構成される．

「被験者へのアンケート」は，レビュー経験に関する質問や，プログラミング経験に関する質問など，いくつかの質問から構成される．

「ソースコード（version1）」は実験のために開発したアプリケーションである．Java で記述されており，内容は GUI アプリケーション（ペイントアプリケーション）である．

†奈良先端科学技術大学院大学情報科学研究科

Graduate School of Information Science, Nara Institute of Science and Technology

実験で被験者に渡した資料では，ソースプログラム（version1）と，ソースプログラム（version2）が，左右に並べられ，変更を加えた部分がハイライトで表示されている．変更は新規機能の追加やリファクタリングなどそれぞれ意味のある変更になっている．被験者には変更を加えたソースコード（version2）に問題がないかどうかを判断してもらい，判断にかかった読解時間を測定する.

本稿では，13 種類のソースコード（version2）の内の 7 種類を，A～C の系統別に分類した．表 1 はそれぞれの差分の概要である．また（1）～（7）はそれぞれの差分内容についての詳細な説明である.

（1） 差分 A1

本差分では，ソースコードの先頭に，プログラムの著作権を示す全 27 行の文字列をコメントアウトして追加した．コメントとソースコードの処理内容には関連性がないため，コメントを追加するに当たり，ソースコードの内容を理解する必要性は低い.

（2）差分 A2

本差分ではクラス名を変更した．この変更によりソースコードが正しく動作するかどうかを判断するには，このクラスの定義部分，このクラスのインスタンスが宣言・生成される部分，および，このクラスを継承しているサブクラスの定義部分を確認する必要がある．このクラスのインスタンスはフィールド領域でのみ宣言・生成されているため，メソッドの内部を確認する必要性はない（メソッド内部で本クラスを参照する部分は存在しない）．また，本差分自体が特定の文字列を機械的に置き換えただけのものであるため，処理内容について理解する必要性は低い．従って本差分により問題が生じるかどうかの判断は変更部分の文法を確認するだけで十分である.

（3）差分 A3

本差分では変数の型を int から float，float から int へと変更した．この変更によりソースコードが正しく動作するかどうかを判断するには，型変更した変数へ値を代入する際にキャストしているか，および，この変数を用いている関数の引数が int，float 両方とも受け付けるかどうかを確認する必要がある．なお，変数への代入は全てマジックナンバーを使用している．マジックナンバーは，1.0 や 2.0 などキャストにより桁あふれや丸め誤差などが生じないことが一目で分かるような値である．従って本差分により問題が生じるかどうかの判断は変更部分の文法を確認するだけで十分である．ソースコードの内容について理解する必要性は低い.

表 1：計 7 種類の差分内容の概要

差分	変更部分の前後を読む必要があるか	変更内容
A1	なし	コメントの追加
A2	なし	クラス名の変更 (メソッド外部の書き換え)
A3	なし	変数の型変更 (メソッド内部の書き換え)
B1	あり	GUI の単純な変更
B2	あり	GUI の単純な変更
C1	あり	GUI の複雑な変更
C2	あり	GUI の複雑な変更

(4) 差分 B1
本差分では GUI 部分を変更した．具体的にはパネル（JPanel）上に並べられたコンポーネントのサイズを変更した．この変更によりソースコードが正しく動作するかどうかを判断するには，パネルの初期化部分でパネル上に配置したコンポーネントの座標が正しく変更されているかどうかを確認する必要がある．なお，変更部分の前後では，各コンポーネントの初期化（プロパティの設定，イベントリスナーの追加など）を順に行っているだけで，特段複雑なロジックが組み込まれているわけではない．

(5) 差分 B2
本差分では GUI 部分を変更した．具体的にはパネル上のドロー領域をクリアするためのボタンを追加した．この変更によりソースコードが正しく動作するかどうかを判断するには，クリアするためのボタンを正しく宣言・初期化しているか，および，ドロー領域をクリアするための処理が適切に記述されているかを確認する．変更部分の前後では，各コンポーネントの初期化（プロパティの設定，イベントリスナーの追加など）を順に行っているだけで，特段複雑なロジックが組み込まれているわけではない．

(6) 差分 C1
本差分では GUI 部分を変更した．具体的にはパネル上のドロー領域内で，新たにブラシ（スプレー状のペン）を用いた描画ができるように，パネル上にブラシ選択ボタンを追加した．この変更によりソースコードが正しく動作するかどうかを判断するには，ブラシ選択ボタンを正しく宣言・初期化されているか，および，ドロー領域にブラシで描画するためのアルゴリズムが正しく組み込まれているかを確認する．なお，変更部分では，各コンポーネントの初期化（プロパティの設定，イベントリスナーの追加など），ブラシによる描画処理などを行っている．また，ブラシによる描画では，多重にネストされたループ内で乱数を用いた座標設定を行っているため，複雑な処理を理解する必要がある．

(7) 差分 C2 本差分では GUI 部分を変更した．具体的にはパネル上のドロー領域内の一部をグラデーションボタンで塗りつぶすための機能を追加した．この変更によりソースコー

ドが正しく動作するかどうかを判断するには，グラデーションボタンを正しく宣言・初期化しているか，および，ドロー領域にグラデーションを用いて塗りつぶすためのアルゴリズムが正しく組み込まれているかを確認する．なお，変更部分では，各コンポーネントの初期化（プロパティの設定，イベントリスナーの追加など），グラデーションによる塗りつぶし処理などを行っている．グラデーションによる塗りつぶしにはシード・フィル アルゴリズムが用いられており，複雑な処理を理解する必要がある．

2.2 実験手順

初めに，被験者にアンケートを実施した．このアンケートには，レビュー経験についての質問が含まれている．レビュー経験に関して「豊富な経験がある」，「多少の経験がある」，「経験がない」の選択肢からどれか一つを選んでもらった．次にソースコード（version1）を理解してもらった．次にソースコード（version1）に変更を加えたソースコード（version2）をレビューしてもらい，変更後のソースコードに誤りがあるかどうかを判断してもらった．その際に，それぞれのソースコードの読解時間を測定した．

3. 実験結果

アンケートで質問したレビュー経験の結果に基づき，被験者を二種類に分類した．以下，レビュー経験に関して，「豊富な経験がある」を選んだ被験者を経験多，「多少の経験がある」を選んだ被験者を経験ありとする．以下，経験多と経験ありの比較結果について示す．

3.1 差分 A1，A2，A3 の実験結果

図 1 は差分 A1，A2，A3 の読解時間の分布である．被験者を経験多と経験ありに分けて記述した．縦軸は分単位の読解時間を表している．それぞれの箱は左端から順に，差分 A1 に対する経験多と経験あり，差分 A2 に対する経験多と経験あり，差分 A3 に対する経験多と経験ありの分布である．

差分 A1（経験多），A1（経験あり）を比較すると中央値は両方とも 3 分と差はなかった．また差分 A2，A3 の経験多，経験ありと比べると読解時間は短かった．

差分 A2(経験多)，A3(経験多)を比較すると読解時間の中央値は両方とも 5 分と差はなかった．一方差分 A2(経験あり)，A3(経験あり)を比較すると，中央値が 7 分から 10 分と増加した．

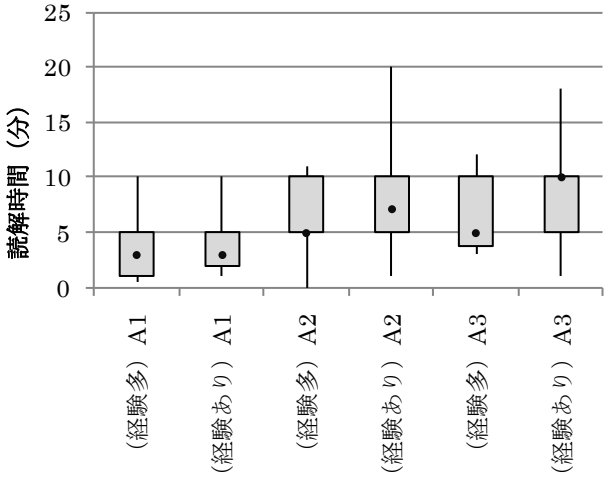

図 1：差分 A1，A2，A3 の読解時間の分布

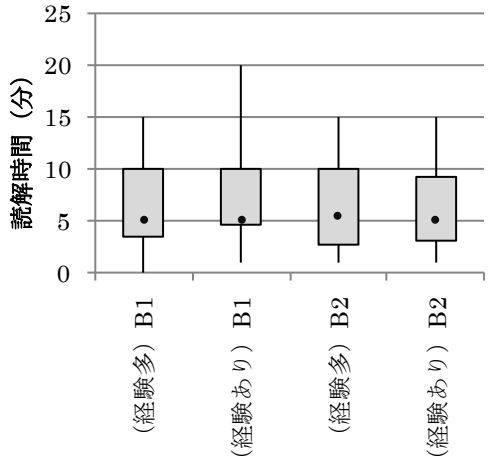

図 2：経験多と経験ありの差分 C1，C2 の読解時間

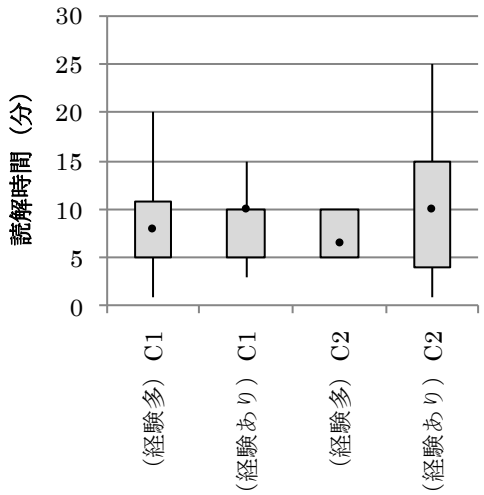

図 3：経験多と経験ありの差分 D1，D2 の読解時間

3.2 差分 B1，B2，C1，C2 の実験結果

図 2 は差分 B1，B2 の読解時間の分布である． 被験者を経験多と経験ありに分けて記述した．縦軸は分単位の読解時間を表している．それぞれの箱は左端から順に，差分 B1 に対する経験多と経験あり，差分 B2 に対する経験多と経験ありの分布である．

差分 B1(経験多)，B1(経験あり)，B2(経験多)，B2(経験あり)の読解時間を比較すると中央値はそれぞれ 5 分，5 分，5.5 分，5 分とほとんど差がなかった．

図 3 は差分 C1，C2 の読解時間の分布である． 被験者を経験多と経験ありに分けて記述した．縦軸は分単位の読解時間を表している．それぞれの箱は左端から順に，差分 C1 に対する経験多と経験あり，差分 C2 に対するが経験多と経験ありの分布である．

差分 C1(経験多)，C1(経験あり)，C2(経験多)，C2(経験あり)を比較すると中央値は順に 8 分，10 分，6.5 分，10 分であった．差分 C1，C2 を比較すると経験多と経験ありの読解時間の中央値の差は C1 の方が差分 C2 に比べて小さかった．また差分 C1，C2 間において，経験多の読解時間の中央値は両方とも同じ値だった．

4. 考察

4.1 文法確認だけで判断できる差分

差分 A1，A2，A3 に共通する点は文法を確認するだけで，ソースコードに問題があるかどうかの判断ができることである．

差分 A1 の読解時間の中央値は差分 A2，A3 に比べ，経験多，経験あり共に低く，また，差分 A1 の経験多，経験ありの間で読解時間の中央値に差が見られない．この原因は，被験者はコメントの内容について理解しようとしておらず，コメントの表記法（文法）に誤りがあるかどうかを機械的に判断しているためと推察される．

差分 A2，A3 で異なる点は，差分 A2 はメソッドの内部に変更が加えられていないのに対して，差分 A3 では変更が加えられていることである．差分 A2，A3 の経験多，経験ありの読解時間を比較すると，経験多の読解時間の中央値は差分 A2，A3 間に差が表れていないのに対し，経験ありの読解時間の中等値は差分 A2 に比べて差分 A3 の方が多くの時間がかかっている．この原因は，経験ありは，差分 A2 を読解する際に，メソッド内の処理内容を理解しなくてもよいと判断したのに対し，差分 A3 ではその判断がなされなかったためと推察される．一方経験多は，差分 A2，差分 A3 共に，読解しなくてもよいと判断したため読解速度に差が表れなかったのではないかと推察される．

4.2 処理内容の理解が必要な差分

差分 B1，B2，C1，C2 に共通する点はメソッド内の処理内容について理解する必要があることである．

図 2 の差分 B1，B2 の経験多と経験ありを比較すると，読解時間の中央値にほとんど差が表れていない．一方，図 3 の差分 C1，C2 の経験多と経験ありを比較すると読解時間の中央値がそれぞれ上昇している．これは差分 B1，

B2 の変更部分は周辺のロジックが単純だったのに対して，差分 C1，C2 には周囲に複雑なロジックが存在したためと考えられる．

以上により，単純な差分では経験の長い被験者とそうでない被験者の間にほとんど差が表れないと考えられ，レビュー経験の長さは読解速度の改善に大きく貢献しない可能性があるといえる．一方，複雑な差分は，レビュー経験の長さが読解速度の改善に貢献するのではないかと考えられる．

5．まとめ

開発経験がソースコードの読解時間に与える影響を調べるために，実務者を対象として実験を実施した．対象ソースコードは実験のために開発したものである．ソースコードのバージョン 2 はバージョン 1 に対して，複数の追加，削除，変更したものであり，新規機能の追加，リファクタリング等，それぞれ意味のある変更となっている．被験者にはバージョン 1 を理解してもらい，各差分によってバージョンアップして問題が起きないかどうかを判断してもらった．差分は，差分の前後を理解しなければ適用可否を判断できないもの，差分だけを読めば適用可否が判断できるものを用意した．

実験結果から，レビュー経験の長い開発者はソースコードを読解する際に，ただ単純に読解するだけではなく，処理内容を理解する必要があるかどうかを判断しながら極力少ない労力で読解しているのに対し，レビュー経験の短い開発者は処理内容を理解する必要があるかどうかの判断がうまくできず，結果，ソースコードの読解に多くの時間がかかっているのではないかと推察される．この問題は，レビュー経験の長いほうが短いほうに比べ，読解時間が短くなっていることから，レビュー経験を積むことにより徐々に改善されていくと考えられる．また一般にコードの読解よりも，変数名の置き換えなどの方が時間的コストが低いことが知られている[3].

また，文法のみで正しく動作するかどうか確認できない差分は，ソースコードが単純な場合は，レビュー経験の長さは読解時間に影響を及ぼさないのに対し，複雑な場合はレビュー経験が長いほどソースコードの読解時間は短くなると考えられる．従ってレビュー経験の長い開発者には複雑なソースファイルを，レビュー経験の短い開発者には単純なソースファイルを割り当てることにより，レビューの効率化につながることが期待される．

今後の課題として，追試実験を行い，実験結果の一般性をさらに高める必要がある．

謝辞

本研究の一部は，文部科学省「次世代 IT 基盤構築のための研究開発」の委託に基づいて行われた．また，本研究の一部は，文部科学省科学研究補助費（若手 B:課題番号 21700033）による助成を受けた．本論文の分析対象データの一部は IBM Academic Initiative Program (http://www.ibm.com/developerworks/university/academicinitiative/)の支援によって収集された．

参考文献

[1] E J. Barry, C F. Kemerer, S A. Slaughter, "On the Uniformity of Software Evolution Patterns" Proceedings of the IEEE 31st International Conference on Software Engineering," pp.106-113,2003

[2] H C. Benestad, B Anda, E Arisholm, "Understanding cost drivers of software evolution: a quantitative and qualitative investigation of change effort in two evolving software systems," Empir Software Eng, Vol.15, No. 2, pp.166-203, 2010

[3] A J. Ko, B A. Myers, M J. Coblenz, H H. Aung, "An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Tasks," IEEE Transactions on Software Engineering, Vol. 32, No12, pp. 971-987, 2006