社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

レビュー開始時における対象物の比較指示によるバグ発見率の向上

應治　沙織†　　上野　秀剛†

† 奈良工業高等専門学校　専攻科
奈良県大和郡山市矢田町 22 番地
E-mail: †a0776@stdmail.nara-k.ac.jp

あらまし　ソースコードを精読しバグを検出する作業であるソースコードレビューは作業者によってその効果が大きく異なる．本稿ではレビュー効率が高い作業者の読み方を，他の作業者に指示することでバグ発見率が向上するか調査する．レビュー効率の高い作業者の読み方を参考に，レビュー開始直後にソースコード全体と設計書を精読した後に設計書とソースコードの整合性を比較するよう被験者に指示する．実験では被験者を 2 グループに分割し，片方のグループにのみ指示した上でコードレビューを行い検出するバグ数を比較する．実験の結果，指示を行ったグループは行わなかったグループに比べ，有意に高いレビュー効率を示し，最大で 18.4%の差が見られた．
キーワード　ソースコードレビュー，レビュー効率，視線，読み方

Improvement of Fault Detection performance with Reading Procedure Instruction in Code Review

Saori OUJI† and Hidetake UWANO†

† Nara National College of Technology
E-mail: †a0776@stdmail.nara-k.ac.jp

Abstract　Fault detection efficiency at careful source code reading (that is, code review) is significantly different between reviewers. In this paper, authors investigate the effect of review procedure instruction which formed from effective reviewers' reading process to reviewers. Our instruction for the reviewer consists of three steps, 1) read design specification at the initial phase of review, 2) read the entire source codes to understand structure of the program, and 3) verify the consistency between the design specifications and the source code. In the experiment, authors divide subjects into two groups and instruct the above steps to a group and compare the fault detection efficiency. The result of the experiment shows that the group with the instruction significantly improved their fault detection ratio.
Key words　Source Code Review，Review Efficiency，Eye Movement，Reading Procedure

1. はじめに

ソースコードレビュー（以後，コードレビュー）とはソフトウェアレビューの一種で，プログラムの中に混入しているバグを検出するために，開発者がソースコードを精読することである．通常，コードレビューは作業者（レビュアー）がプログラムをコンパイルや実行すること無しに，画面上，あるいは紙に印刷したソースコードに対して行う．ソフトウェア開発において結合テストやシステムテストのような後工程で誤りが見つかった場合，その除去には多くの労力とコストがかかる．そのため，実行可能なコードが存在しない，開発初期に実施可能なコードレビューは誤りの早期発見と開発コストの削減に有用である．

ソフトウェアレビューの効率を向上させるために，これまでに多くの研究が行われ，様々な手法が提案されている [1]〜[3]．これらの手法はいずれもレビュー対象のドキュメントに対して，どのような基準で読み進めるか，またはどのような特徴に着目して読むかといったレビュー基準を定めることで，レビューの効率・効果を高める点に特徴がある．一方で，従来研究の結果を比べると，同じ手法を用いたレビューにおいても，その効果・効率には研究ごとに大幅な差が見られる．これは，従来研究で提案されているレビュー手法が大まかな方針のみを指示しているため，被験者によってその解釈や理解度が異なってしまい，レビュー手法の意図通りにレビューした被験者とそれ以外の被験者でレビュー効率に差が出たと考えられる．

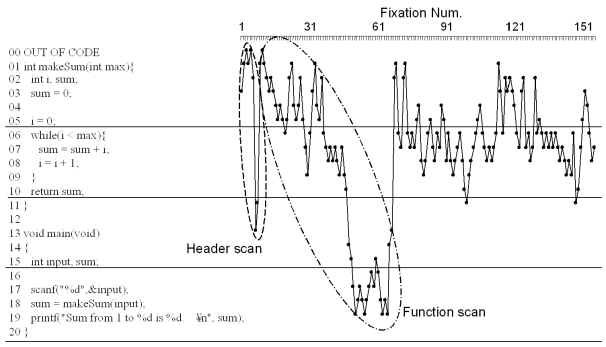

図 1　レビュー作業者の視線移動

そこで，本研究では被験者間で解釈の違いが起きないような，具体的な手順を指示してレビューを実施してもらう．著者の一人が行った過去の研究で得られた，レビュー効率の高いレビュアーの読み方を元に作成した手順を被験者に指示することでレビュー効率が向上するか調査する．実験では読み方を指示するグループと，指示しないグループにそれぞれレビューを実施してもらい，レビュー効率や指摘精度を比較する．その際，読み方を指示したグループが，指示した読み方を実施しているかを確認するためにレビュー中の視線移動を計測する．

本論文の構成は以下のとおりである．2. 章では関連研究について示す．3. 章ではソフトウェアレビューにおける読み方の指示について説明する．4. 章では実験環境や実験手順などを示し，5. 章では実験結果と考察を述べる．6. 章ではまとめと今後の課題について述べる．

2. 関 連 研 究

2.1 ソフトウェアレビュー

ソフトウェアレビュー手法は CBR や PBR，また PBR の一種であり，ユーザの視点からレビューを行う Usage-Based Reading(UBR) など，これまでに多数提案され，その性能が比較されている [1]〜[3]．松川らはソフトウェアの仕様記述言語である UML で記述された設計書を対象としたレビュー実験で，CBR と PBR を比較している [4]．実験の結果は PBR が図と図の関連性や一貫性に関する意味的なバグを発見しやすく，CBR は構文的なバグが発見しやすいことを示している．

一方で，手法の性能を比較している研究の結果を比べると研究によって異なる結果を示している場合がある．例えば UBR と CBR を比較した研究では Porter らは UBR のほうが優れている [5] とする一方で，Lanubile らは有意な差はない [6] としており，手法の違いについて明確な結果が得られていない．また，手法による差よりも同じ手法を用いた被験者間の差が大きい [7] ことから，レビュー方法の大まかな方針を指示するだけではレビュー効率の向上には必ずしも繋がらない恐れがある．本研究では，レビュー開始直後における提示物の読み方についてより具体的な指示をすることでレビュー効率が向上するか調査する．

2.2 視線計測による能力評価

作業者の視線移動を計測し，行動の分析や能力を比較する研究は認知科学の分野においてよく行われている [8], [9]．村田らは自動車運転時の危険予知における熟練者と初心者の視線の動きを比較している [8]．自動車運転状態の静止画を運転熟練者と初心者に見せ，危険を予知する際の視線運動を計測する実験の結果，危険度の高いカーブで熟練者と初心者の視線移動の違いが顕著になる一方で，直線道路や交通量の少ない道路では熟練者と初心者の視線移動の違いが少なくなった．Law らは腹腔鏡手術の訓練装置を使用している初心者と熟練者の視線移動を分析し，熟練者は初心者に比べて患部を集中して見ていることを明らかにしている [9]．

ソフトウェア開発の分野においても，視線計測による開発者の分析や支援に関する研究が行われている [10]〜[13]．中道らは Web ページ閲覧時のユーザビリティ評価に視線情報を利用している [11]．Web ページのユーザビリティの評価指標に実験時に記録した視線データを用いることで，Web ページのユーザビリティについてより多くのコメントが得られることを明らかにした．Stein らはソフトウェアのデバッグ被験者に他の被験者の視線移動を見せることが作業の助けになるということを示している [12]．分析結果から，他の被験者の視線移動を見てからデバッグを開始した被験者は視線移動を見なかった被験者よりも早くバグを発見できることを明らかにした．

著者の一人はこれまでにレビュアーがレビュー中にソースコードのどこに注目しているか視線移動から計測し，レビュー能力との関係について調べている [13]．被験者に仕様書，設計書，ソースコード，コードチェックリストを提示した実験の結果，レビュー開始時にコード全体を上から下に向かって眺める動作を行わない被験者は誤り検出までの時間が長くなる傾向が見られた．本稿では，これまでの研究結果を基に，レビュー開始時にプログラム全体を見渡すよう指示することでレビュー効率が向上するか調査する．

3. コードレビューにおける読み方の指示

本研究ではコードレビューにおけるレビュー対象物の読み方を作業者に指示することでレビュー効率が向上するか調査する．図 1 に著者らの過去の研究で計測した，レビュー効率が高い作業者の視線の動きを示す．図の左側はレビュー対象のソースコードを，右側がレビュー開始からの視線移動を示している．図は作業者が開始直後にソースコードに含まれる 2 つの関数のヘッダー部分と各関数を見渡していることを示している．レビュー効率が高い他の作業者の視線移動からも同様の傾向が見られた．

バグの検出にはプログラムの制御フローやソースコード上のクラス・メソッドの記述位置といった構造を理解する必要がある．著者らの過去の研究で見られた視線移動はプログラムの構造をレビュー開始時に確認する動作であると考えられる．レビュー開始時にプログラムの構造やクラス・メソッドの記述位置を把握することは，それ以降のレビューを効率的に実施するために有用である．また，設計書を含んだコードレビューの場合，レビュー対象であるソースコードだけでなく，設計書に記述されたプログラムの機能や構造の理解も必要である．設計書を読むことでプログラム自体の理解が効率的になることに加え，

設計書とソースコードのずれを見つけることで，効率的にバグを発見できると考えられる．

そこで，本研究では作業者にコードレビュー開始時の読み方を指示し，1) ソースコードの構造を理解し，2) 設計書の内容を把握することで，レビュー効率が向上するという仮説を立てる．また，加えて 3) 設計書とソースコードの整合性を確認するよう読み方を指示することで，レビュー効率が向上するか確認する．以下に作業者に与える指示を示す．

（1） 設計書をよく読んでください．
（2） コード全体にざっと目を通してください．
（3） 設計書に書かれているフィールド，メソッドが正しくソースコードに実装されているか確認してください．

手順 (1) は設計書を読むことでプログラムの設計を理解してもらうための指示である．手順 (2) と (3) はソースコードの構造を理解してもらうとともに，ソースコードと設計書の整合性を確認するための指示である．本章で述べてきたとおり，これらの手順を指示することで作業者のレビュー効率が向上すると考えられる．一方で，指示を実施することでレビュー作業開始から数分間はバグを検出しなくなり，指示がない場合と比べてレビュー効率が低下する可能性がある．本稿では，指示した作業者と指示しなかった作業者のレビュー効率を比較し，レビュー時の読み方指示の効果を調査する．

4. 実　　験

レビュー対象物の読み方を被験者に指示することでレビュー効率が向上するか評価するために被験者実験を行う．実験では被験者が指示した読み方を実施しているか確認するために被験者の視線を計測する．

4.1 計測する指標

レビューの性能を評価するために，実験では以下の 3 つの指標を計測する．

- 指摘回数
- 指摘精度
- バグ発見率

指摘回数は被験者がレビュー中に行ったバグの指摘の回数を示す．バグを含むと思われる行に対して指摘してもらい，同じ行に連続して指摘があった場合は連続した回数をカウントする．また，実験時間中に指摘の取り消しがあった場合，指摘回数には含めない．

指摘精度は被験者の指摘のうち，バグの症状や原因について正しく記述された指摘の割合を示す．指摘の正否については実験者が個々に判断する．

バグ発見率はソースコード中に埋められたバグのうち，被験者が指摘した割合を示す．本稿で行った実験では，著者らが埋めたもの以外に不具合の指摘はなかったため，事前に埋められた不具合全てを検出した場合に 100%とした．

4.2 実 験 環 境

本研究では，著者らがこれまでに作成した視線計測環境 [13] を用いてレビュー中の視線移動を計測する．本環境は，視線計測装置と視線計測装置の制御用 PC 1 台と，コードレビュー実験用ソフトウェアを内蔵する実験用 PC 1 台から構成される．視線計測装置は内蔵カメラで撮影した被験者の眼球映像から，ディスプレイ上の注視している座標点を算出する．

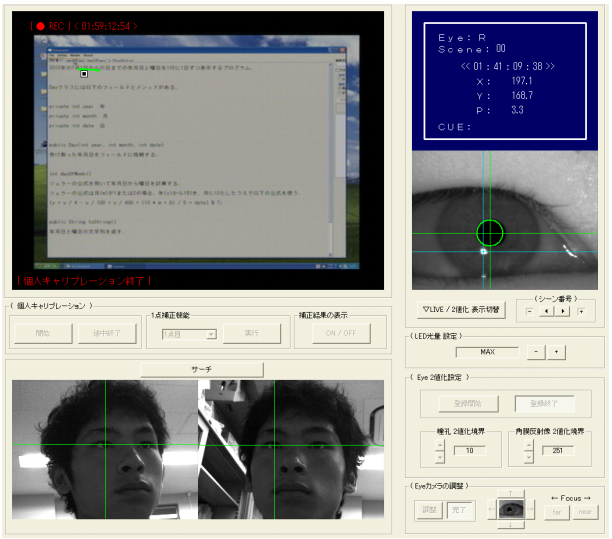

図 2　EMR-AT Voxer Basic+

本研究ではソースコードを精読する被験者の視線移動を計測するため，被験者の視線の動きを行単位で精密に識別する必要がある．そこで本環境では視線計測装置に，高解像度で精度の高い計測が可能である nac Image Technology 社の非接触アイマークレコーダ EMR-AT Voxer Basic+を用いる．本計測装置は被験者に装置や計測用のめがねなどを装着する必要が無いため，被験者に負担をかけることなく計測できる．

コードレビュー実験用ソフト Crescent (Code Review Evaluation System by Capturing Eye movemeNT) は視線計測装置が出力するディスプレイ上の座標情報から，被験者が注視しているレビュー対象文書の行を特定する．Crescent は被験者にレビュー時に閲覧するソースコードや他の文書（設計書やチェックリスト）をディスプレイに表示すると共に，被験者のウィンドウ移動や表示する文書の切り替え，スクロールなどの操作を計測する．操作履歴からディスプレイに表示されている文書の行とその座標を算出し，視線計測装置が出力する座標情報から被験者がどの文書のどの行を見ているか計算する．また，レビュー中に誤りを発見した行をダブルクリックすると，誤り報告用のウィンドウが表示され，誤りの内容と位置を記録できる．

実験では上記の構成を 2 組用意し，被験者 2 名を同時に計測した．本計測装置で視線を計測している様子を図 2 に示す．また，ヘッドレストがついた椅子を用いることでレビュー中に被験者の頭が動かないよう配慮した．

4.3 実 験 手 順

被験者を 2 つのグループ（指示有り，指示無し）に分け，片方のみに指示を与えたうえで Java で書かれたソースコードを対象としたレビューを行ってもらい，それぞれのレビュー効率を計測する．2 つのグループ間で被験者の能力差が現れないように，予備実験で全被験者のレビュー効率を計測し，バグ発見

表 1　実験で提示するプログラム

	仕様	設計書	ソースコード	バグ数
予備実験 練習	整数 n を入力すると 1 から n の和を表示する.	12 行	32 行	1
予備実験 (1)	7 月 1 日から 31 日までの年月日と曜日を 1 行に 1 日ずつ表示する.	25 行	47 行	5
予備実験 (2)	複数の銀行口座の残高を表示する.	37 行	78 行	6
本実験 練習	n を入力すると n 段のピラミッドをアスタリスクで表示する.	30 行	25 行	1
本実験	ファイルに記録された賃貸物件から条件に合う物件を検索し表示する.	30 行	138 行	7

率が等しくなるようグループを分割する．実験の被験者として，Java によるプログラミングの経験が 1 年以上ある学生 15 人を対象とする．

4.3.1　予 備 実 験

予備実験では，表 1 に示すプログラムを対象に被験者にレビューを実施してもらう．いずれも Java で記述されたソースコードと，日本語で記述された設計書からなる．ソースコードにのみ，著者らが複数の誤りを混入した．誤りはソースコード 10 から 20 行につき 1 件を混入し，ソースコードの長さからバグの数を予測できないよう配慮した．誤りは，ボーリス・バイザーの作成したバグ分類 [14] に基づいて，発生頻度が高く，特定のプログラム言語に依存しない以下の 3 種類を埋めた．

- 機能不良： 設計とソースコードの相違に起因する誤り
- 構造不良： 制御フローやアルゴリズムに起因する誤り
- データ不良： データの仕様や形式，種類，初期値に起因する誤り

全てのレビューにおいて，著者らが作成したコードレビュー用チェックリストを被験者に提示する．チェックリストには，ソースコードの完全性，初期化，メソッド呼び出し，演算，データ・ファイル，制御について正しくコーディングされているかを確認するような項目を用意した．

予備実験では全ての被験者に 4.2 節で示した計測環境を用いて設計書，ソースコード，チェックリストを読み，ソースコードに含まれる誤りを検出するよう指示する．レビュー時間には制限を設定せず，被験者が誤りをすべて発見した，またはこれ以上の誤りを発見できないと判断したときに終了するよう指示する．誤りを検出した際は，誤りが含まれている箇所をレビュープラットホームの機能を用いて被験者に記録させる．検出した誤りの正否はレビュープラットホームが出力する操作履歴をともに著者が判断する．すべての被験者は最初に練習タスクを実施した後に，2 つのプログラムをレビューする．2 つのプログラムをレビューする順序については，学習効果を考慮して半数の被験者はタスク 1 を最初に実施し，残りの半数はタスク 2 を最初に実施する．

4.3.2　本　実　験

本実験は被験者を 2 つのグループ（指示有り，指示無し）に分割し，レビューを実施してもらう．

表 1 に本実験で被験者に提示するプログラムを示す．レビューに用いる環境や混入するバグの種類や数，検出したバグの記録方法については予備実験と同様とする．指示有りグループにはタスク開始前に 3. 章に記した指示を口頭で行う．本実験では，各グループの読み方を確認するために被験者の視線を計測する．

5.　結果と考察

5.1　視 線 移 動

図 3 に本実験における” 指示有り” グループの被験者 1 名の視線移動を，図 4 に” 指示無し” グループの被験者 1 名の視線移動を 1200 秒まで示す．図の横軸はレビュー開始からの経過時間（秒）を，縦軸は注目している提示物のブロックを示している．ブロックは提示物に書かれた内容を基準に，1 行から 23 行程度の意味のまとまりで分割した．

図 3 から，指示有りの被験者はレビュー開始直後から設計書，ソースコードを順に読み，その後，設計書とソースコードを交互に移動しながら読んでいることがわかる．この被験者は，開始 480 秒経過時には設計書の”Main クラス main()” ブロックの内容を確認した後，それに対応している Main.java ファイルの”main() ファイル読み込み” ブロックを読んでいることが確認できる．これは，被験者が指示通りにレビュー開始直後にまず設計書とソースコードを精読し，その後，設計書の内容とソースコードを比較していることを示している．指示有りグループの他の被験者においても同様の視線移動が見られた．

図 4 から，指示無しの被験者は Main.java 以外の提示物をあまり読んでおらず，各提示物の全体を見渡す動作も行っていないことがわかる．指示無しグループの他の被験者においても同様に，レビューの初期に設計書を読んでおらず，提示物全体を見渡す動作もほとんど見られなかった．このことから指示無しグループはプログラムの内容や設計書の構造を把握せずにレビューを行っていると考えられる．

5.2　バグ発見率

本実験における指示無しグループのバグ発見率は平均で 60.7%だったのに対し，指示有りグループは 67.3%と 6.6%の差が見られた．これは，レビュー開始時の読み方を指示することで，指示無しの場合と比べてプログラムの設計や構造に対する理解が高くなったためと考えられる．2 つのグループに対して Welch の t 検定（両側検定）を行った結果，有意な差は見られなかった．

図 5 に指示有りグループと指示無しグループのレビュー時間とバグ発見率の遷移を示す．図の横軸はレビュー開始からの経過時間（分），縦軸はバグ発見率を表している．レビュー開始から 10 分までの間，指示有りは指示無しと比べてバグ発見率が低い．それ以降，10 分から 20 分までは両グループでバグ発見率に差は見られないが，20 分以降では指示有りの方が指示無しに比べてバグ発見率が高く，レビュー開始後 26 分の時点で 18.4%の差が見られた．指示有りと指示無しの分ごとのバグ発

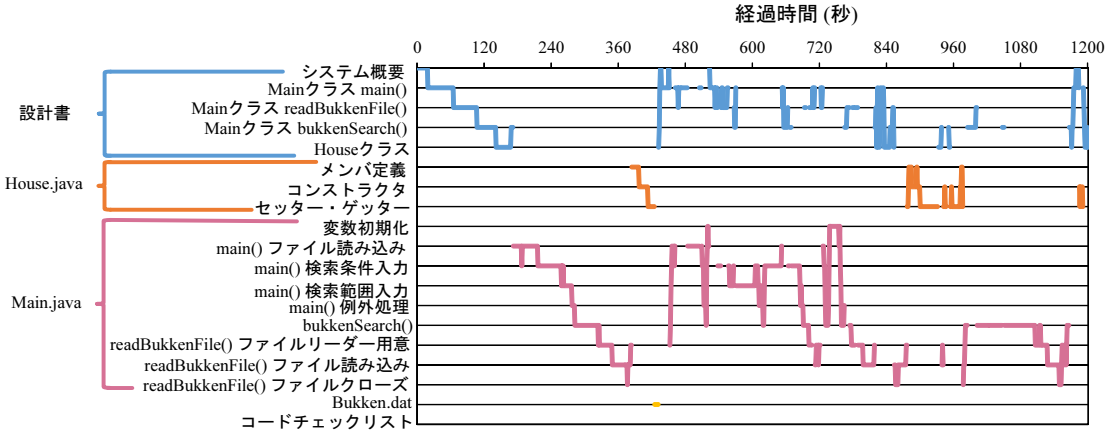

図 3　指示有りグループの視線移動

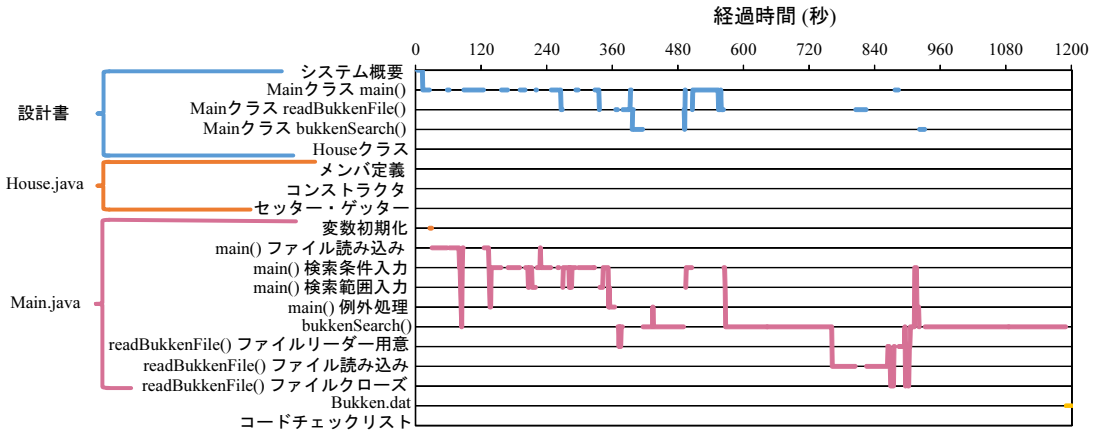

図 4　指示無しグループの視線移動

見率の差に対して Welch の t 検定（両側検定）を行った結果，25 分から 30 分の範囲において，$p < 0.05$ で有意な差が見られた．

指示有りの被験者は指示にしたがって設計書を読んだり，ソースコードの構造を確認していたため，レビュー開始から 10 分間はバグをほとんど検出できなかったと考えられる．しかし，それ以降はプログラムの設計やソースコードの構造を理解した上でレビューしたため，指示無しのグループに比べて効率的にバグを指摘できたと考えられる．ソースコードをレビューする場合，レビューにかける時間は最大でソースコード作成にかかった時間の半分とも言われており（注1），短時間で実施されるわけではない．したがって，本実験で対象とした 138 行のプログラムに対して 25-30 分の範囲において有意にバグ発見率が向上したことには一定の価値があるといえる．行数が多いソースコードが対象の場合，プログラムの理解には設計書の内容やソースコードの構造理解がより重要になるため，本稿で提案した指示はより効果的になると考えられる．

5.3　指摘精度

図 6 に各グループのバグ指摘回数を，図 7 に指摘の精度を時間帯ごとに示す．レビュー開始から 30 分まで両グループの指摘

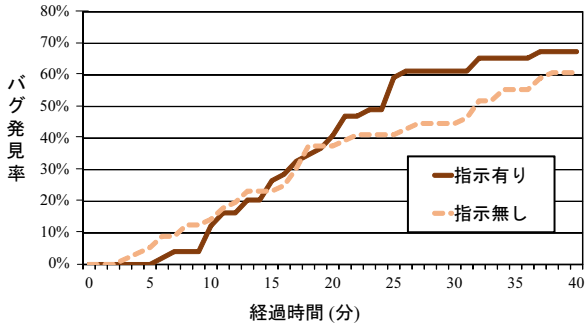

図 5　バグ発見率の推移

回数にほとんど差が見られない一方，11 分から 20 分の指摘精度では指示有りグループが 18.3%，21 分から 30 分で 47.2%高く，読み方の指示によって被験者の正しい指摘が増えたといえる．

また，指摘回数と指摘精度の推移は，指示有りグループの指摘回数と指摘精度が開始 31 分以降に指示無しグループより低くなっていることを示している．図 5 に示されたとおり，指示無しグループのバグ発見率がレビュー終了時点（40 分）で 60%であったのに対して，指示有りグループは 25 分で同じバグ発見率に到達し，それ以降は大きな変化がなかった．これは，指示有

（注1）：http://msdn.microsoft.com/ja-jp/library/ms182019.aspx

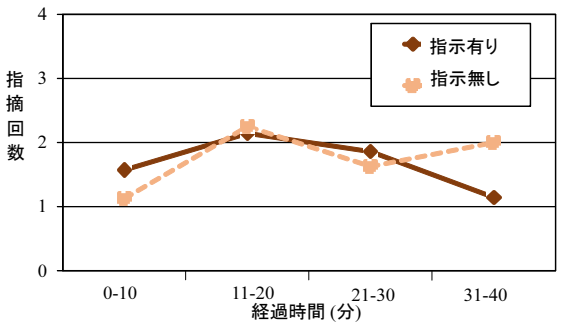

図 6　バグの指摘回数

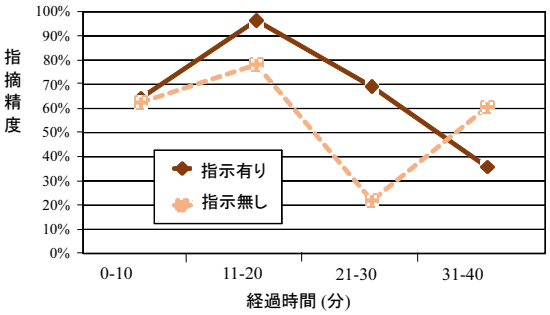

図 7　バグの指摘精度

りグループの被験者が 25 分時点でソースコードをレビューし終え，以降の時間では指摘ができず，有効な指摘もできなかったことを示唆している．したがって，読み方の指示は作業者が正しい指摘をレビューの早い段階で行うために有効であるといえる．

6. おわりに

本稿ではソースコードレビューにおいて，作業者に対しレビュー開始時に設計書を読むこと，ソースコード全体を見渡すこと，設計書とソースコードの整合性を確認することを指示することで，レビュー効率が向上するか検証した．実験の結果，指示をしたグループのバグ発見率はレビュー終了時点で 6.6%高かった．また，時間ごとの推移から指示有りグループのバグ発見率は 25 分から 30 分の範囲で有意に高く，最大 18.4%の差が見られた．これらの結果から，作業者にレビュー開始時の提示物の読み方を指示することは，レビュー効率の向上に有用と考えられる．

本稿で用いた指示は事前準備や訓練をすることなく実施できる簡単なものであるため，開発現場への導入が容易で，特にレビュー経験の浅い作業者に対して有効と考えられる．したがって，ソフトウェア開発組織における新人研修や大学などの講義といった，初学者を対象とした教育において効果があると推測できる．一方で，レビュー経験が豊富な熟練者は既に実施している作業であるため，効果が得られにくいとも考えられる．今後，作業者の経験やレビュー対象物に対する事前知識による効果の違いを調査するとともに，熟練者に有効な指示を調査していく．また，本稿ではソースコードレビューを対象としたが，設計書を対象としたレビューにおいても要求仕様書を読んでから設計書をレビューするような指示は有用であると考えられる．

また，本稿で実施した指示内容はレビュー対象物の精読を始める前に，その関連文章である設計書の内容を理解させている点から，レビュー前に実施する準備フェーズと類似している点がある．一方で，本稿の実験では指示内容を実施している間にもバグの検出が可能であるため，厳密には準備フェーズとは異なる作業といえる．今後，同様の作業をレビュー時間外に実施した場合とレビュー作業の一部として実施した場合のレビュー効率や効果を分析してくことで指示の効果を明確にする．

謝辞　本研究の一部は JSPS 科研費 若手研究 (B) 24700038 の助成を受けて行われた．

文　献

[1] M. E. Fagan : “Design and Code Inspection to Reduce Errors in Program Development,” IBM Systems Journal, Vol.15, No.3, pp. 182-211, (1976).
[2] F. J. Shull : “Developing Techniques for Using Software Documents: A Series of Empirical Studies,” Ph.D. thesis, Univ. of Maryland, (1988).
[3] T. Thelin, P. Runeson, and B. Regnell : “Usage-based Reading – An Experiment to Guide Reviewers with Use Cases,” Information and Software Technology, Vol.43, No.15, pp. 925-938, (2001).
[4] 松川文一, Giedre Sabaliauskaite, 楠本真二, 井上克郎 : “UML で記述された設計仕様書を対象としたレビュー手法 CBR と PBR の比較評価実験”, オブジェクト指向最前線 2002, pp. 67-74, (2002).
[5] A. A. Porter, L. Votta : “Comparing Detection Methods for Software Requirements Inspection: A Replication using Professional Subjects,” Empirical Software Engineering, Vol.3, No.4, pp. 355-380, (1988).
[6] F. Lanubile, G. Visaggio : “Evaluating Defect Detection Techniques for Software Requirements Inspections,” ISERN Report no. 00-08,pp. 1-24, (2000).
[7] T. Thelin, P. Runeson, and C. Wohlin : “An Experimental Comparison of Usage-Based and Checklist- Based Reading,” IEEE Transaction on Software Engineering, Vol. 29, No. 8, pp. 687-704, (2003).
[8] 村田厚生, 森若誠 : “危険予知課題における運転者の視覚情報処理特性-運転初心者と運転熟練者の比較 ”, 人間工学,Vol.46,No.6, pp. 393-397, (2010).
[9] B. Law,M. S. Atkins,A. E. Kirkpatrick,,A. J. Lomax, and C. L. Mackenzie : “Eye gaze patterns differentiate novice and expert in a virtual laparoscopic surgery training environment,” In Proceedings of ACM Symposium of Eye Tracking Research and Applications (ETRA), pp. 41-48, (2004).
[10] S. Zhai,C. Morimoto,and S. Ihde : “Manual and gaze input cascaded (MAGIC) pointing,” In Proceedings of The SIGCHI Conference on Human Factors In Computing Systems ‘99, pp. 246-253, (1999).
[11] 中道上, 島和之, 中村匡秀, 松本健一 : “視線を利用した Web ユーザビリティ評価環境”, 情報処理学会論文誌,Vol.44,No.11,pp. 2575-2586, (2003).
[12] Randy Stein,Susan E. Brennan : “Another Person’s Gaze as a Cue in Solving Programming Problems,”In Proceedings of The 6th International Conference on Multimodal Interface, pp. 9-15, (2004).
[13] Hidetake Uwano : “Measuring and Characterizing Eye Movements for Performance Evaluation of Software Review”, Ph.D. thesis, Nara Institute of Science and Technology, (2009).
[14] Boris Beizer : “ソフトウェアテスト技法”, (1994).