電子情報工学専攻

Advanced Electronic and Information Engineering Course

平成27年度 専攻科特別研究論文

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

Performance Improvement for Source Code Review with Reading Procedure Instruction

指導教員名　上野　秀剛　内田　眞司

論文提出者名　應治　沙織

独立行政法人 国立高等専門学校機構

奈良工業高等専門学校 専攻科

National Institute of Technology, Nara College

Faculty of Advanced Engineering

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

Performance Improvement for Source Code Review with Reading Procedure Instruction

応治　沙織
Saori Ouji

独立行政法人 国立高等専門学校機構
奈良工業高等専門学校 専攻科 電子情報工学専攻
大和郡山市矢田町 22 番地（〒 639-1080）
National Institute of Technology, Nara College,　Faculty of Advanced Engineering
22 Yata-cho, Yamatokoriyama, Nara 639-1080, Japan

Abstract— Source code review is a method that read source codes under developing to detect defects. The methods ' efficiency is strongly influenced by performance of the developers. Our previous study showed that higher performance developers ' reading pattern during code review is different from low performance developer. In this pa- per, authors measure an effect of reading pattern instruction that the high performance developer execute. The instruction is constructed based on the analysis of the higher performance developers ' eye movement; Reviewer read an entire of source code and design documents at the beginning of code review, then verifies a consistency between design document and source code. In the experiment, authors divide subjects into two group, give the instruction for one of the groups. The effect of the instruction is evaluated by comparing the fault detection ratio between the two groups. The result of the experiment showed that fault detection ratio of the instruction group is significantly higher than the non-instructed group; the fault detection ratio of instruction group was 17.1% higher at most.

Keywords— software review, code review, fault detection, eye movement analysis

関連業績リスト

1. 應治 沙織, 上野 秀剛, “コードレビュー時の読み方教示によるレビュー効率の向上”, 情報処理学会研究報告　ソフトウェア工学研究会, Vol.2014-SE-185, No.2, pp.1-8, 2014 年 7 月 9 日
2. 應治 沙織, 上野 秀剛, “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”, 電子情報通信学会　教育工学研究会, 4 Vol.2015-01-ET, pp.1-6, 2015 年 1 月 31 日

目次

図目次

表目次

1. はじめに

ソフトウェアレビューとはソフトウェア開発においてプログラム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や設計書を精読する工程である．コードレビューは，ソフトウェアレビューのひとつで，ソースコード中に混入しているバグを見つける目的で開発者がソースコードを精読する工程を指す．通常，コードレビューは作業者（レビュアー）がプログラムのコンパイルや実行なしに，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システム内に誤りが見つかった場合，その除去には開発の初期工程で見つかった場合に比べ多くの労力とコストがかかる．したがって，開発の早い段階で可能な限り多くのバグを見つけ，修正することが必要となる．コードレビューは実行可能なコードが存在しない開発初期であっても実施できるため，バグ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 Checklist-Based Reading や　Perspective-Based Reading などの手法が提案されている．従来手法にはいずれも，レビュー対象のドキュメントに対して，読み方の基準や着目する特徴を定めることでレビュー効率を高める特徴がある．しかし，同じ手法を適用したレビュー実験においても，そのレビュー効率には個々の実験間で大きな差が見られる．実験間で見られる効率の差は，従来のレビュー手法が読み方の大まかな方針のみしか指示しないため，個々の被験者によってその解釈や理解度が異なってしまうことが原因と考えられる．その結果，手法の意図通りにレビューを行った被験者とそれ以外の被験者の間で，レビュー効率に差が現れたと考えられる．

本研究では被験者間で解釈の違いが起きないように，具体的な手順を指示して，レビューを実施してもらう．レビュー手順（読み方）は，先行研究で行った視線分析によって得られた，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用いる [12]．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 章では論文全体をまとめる．

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とはソフトウェア開発においてプログラム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や設計書を精読する工程である．具体的な手法として，Checklist-Based Reading(CBR) や Perspective-Based Reading(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) などがこれまでに提案されており，その性能が比較されている [1, 2, 3]．多くの手法が提案されている一方，同じ手法を適用した複数のレビュー実験において，実験間で効率に差が見られるなど，それぞれの手法の性能評価は必ずしも統一されていない．例えば，UBR と CBR を比較した場合，Porter らが UBR のほうが優れている [4] とする一方，Lanubile らは有意な差はない [5] としており，手法間の効率差について一貫性のある結果が得られていない．同じ手法を適用したレビュー実験においても，そのレビュー効率には個々の実験間で大きな差が見られる．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 [6] 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，ソフトウェアレビューのうち，特にコードレビューに着目し，レビュー開始直後における提示物（ソースコードや設計書など）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

認知科学の分野において，特定の作業時の視線移動を計測・分析することで，作業者間の行動や能力の差を比較する研究が行われている．例えば，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，熟練者が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 [8]．

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．中道らは，計測した視線データを Web ページのユーザビリティ評価指標として用いることで，そのページのユーザー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 [10]．Stein らは，デバッグ開始時に他の開発者の視線移動を見た被験者が，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示した [11]．上野らは，コードレビュー時の視線移動の計測から，効率の良いレビュアーがソースコードのどこに注目するか調べている [12]．これらの研究から視線移動は作業時における作業者の行動を分析するために有効であると言える．本研究ではレビュー時の作業者の視線移動を計測することで指示が与えた作業者の読み方の違いを計測する．

3. コードレビューにおける読み方の教示

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被験者に指示することでレビュー効率が向上するか，視線計測を用いて調査する．上野らの研究ではレビュー効率の高い作業者はレビュー開始時に提示物全体を見渡す動作を行うことを明らかにした．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．上野らの研究で見られたレビュー効率が高い作業者らの視線移動は，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化することに加え，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．従来手法では，例えば CBR でも，プログラムを読む際の注意点について言及されている．CBR では，「設計された内容は完全にコーディングされているか」と言ったコードの完全性に関する項目や，「＝，＝＝，｜｜などの演算子は正しく使われているか」と言ったプログラミング言語に依存する項目がある．しかし，いずれの項目についても具体的にどう読めば「設計された内容は完全にコーディングされているか」，「＝，＝＝，｜｜などの演算子は正しく使われているか」確認できるか書かれておらず，結局，読み方がレビュアーによって変わり，レビュー効率にも差が出る．そこで本研究では読み方に個人差が生じない読み方をレビュアーに指示する．

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，プログラムへの理解が深まり，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．上記の 3 点を明確に被験者へ伝えるため，レビュー開始時の被験者に次の指示を行う．

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．これらの手順を教示することで，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，手順を実行している間バグを指摘する作業を行わなくなり，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，これらの手順を教示した被験者グループと，しなかった被験者グループのレビュー効率を比較することで，手順の教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 点を計測する．

- 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，バグの症状や原因について記述してもらう．同じ行に対する指摘が連続した場合は一度の指摘としてカウントする．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集計時に著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，上野らが作成した視線計測環境 [12] を用いてレビュー中の視線移動を計測する．本研究で使用した実験環境は，視線計測装置とその制御用 PC 1 台と，コードレ

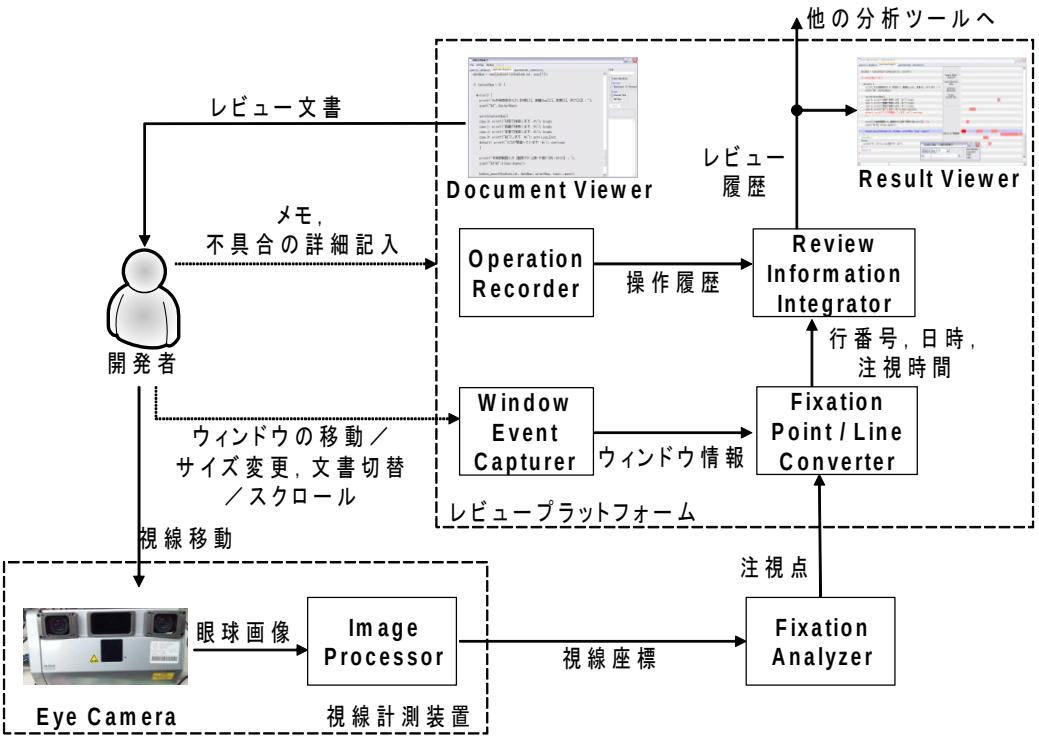

図 4.1　実験環境

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．レビュー対象となる提示物は実験用 PC のディスプレイ上に表示される．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．実験環境を図 4.1 に具体的に示す．

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．EMR-AT Voxer Basic+ は検出分解能 0.3 °，検出レート 60Hz で計測する．計測誤差は当環境においてディスプレイ上で約 5.4pixel となる．これはソースコードを表示したときに，約 0.45 行に相当する．本計測装置で視線を計測している様子を図 4.2 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．ソースコード内の誤りは，ボーリス・バイザーの作成したバグ分類 [13] 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

4.4 予備実験

予備実験では，表 4.1 に示すプログラムの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り，ソースコードにのみ，著者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．

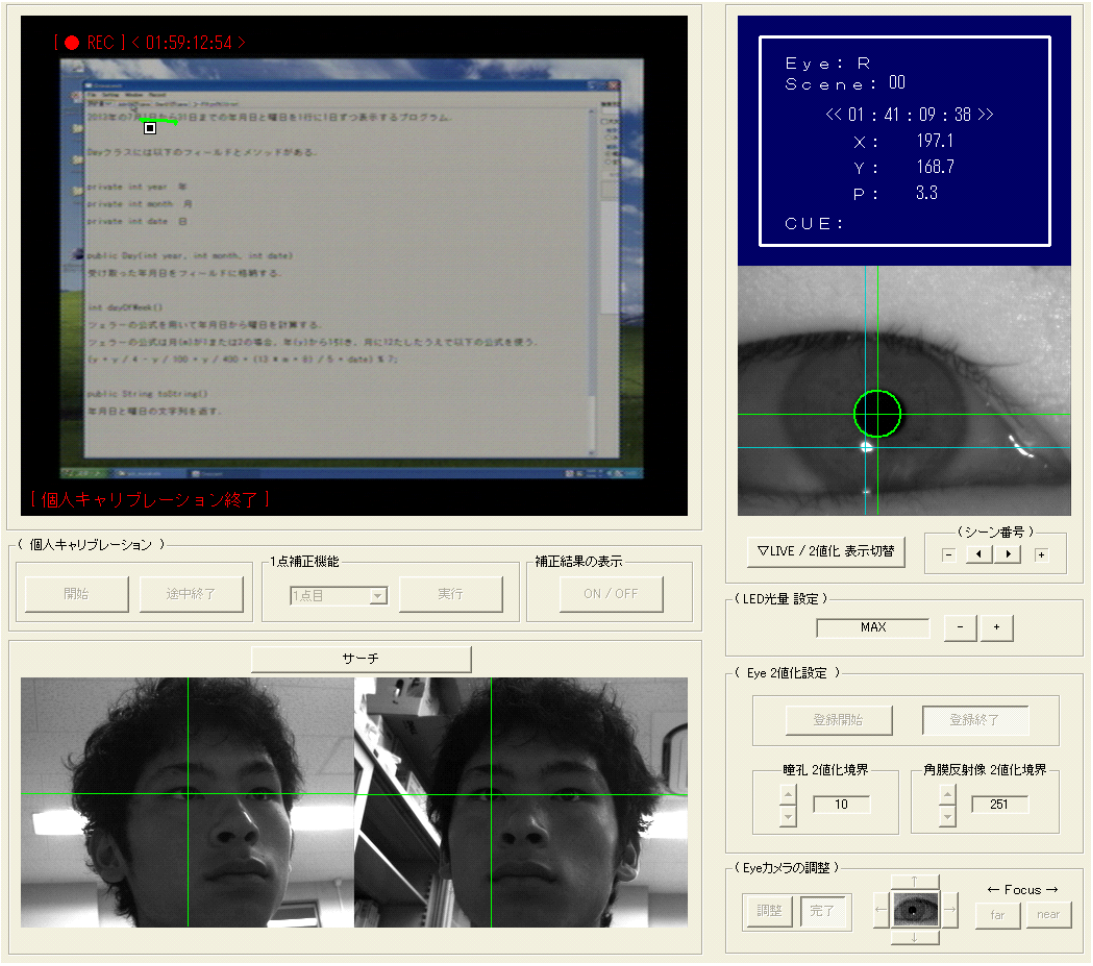

図 4.2　EMR-AT Voxer Basic+

また，全てのレビューにおいて，上野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 [12]．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．

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

表 4.1　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

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．さらに，すべての被験者は最初に練習タスクを実施した後に，予備実験 (1)，予備実験 (2) をレビューする．2 つのタスク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予備実験 (1) を最初に実施し，残りの半数は予備実験 (2) を最初に実施する．ここで学習効果とは，予備実験を行うことで生じるレビューへの慣れを意味する．

4.5　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 4.1 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 5.1 に本実験における指示有りグループの被験者 1 名の開始から 1200 秒までの視線移動を示す．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．ブロックとは提示物に書かれた内容を基準に，1 行から 23 行程度の意味のあるまとまりで分割したものである．図 5.1 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．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 5.1 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」ブロックのデータ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．

図 5.2 に指示無しグループの被験者 1 名の視線移動を示す．図 5.2 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有りグループのバグ発見率は平均で 73.5% だったのに対し，指示無しグループは 66.1% と，両グループの間で 7.4% 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

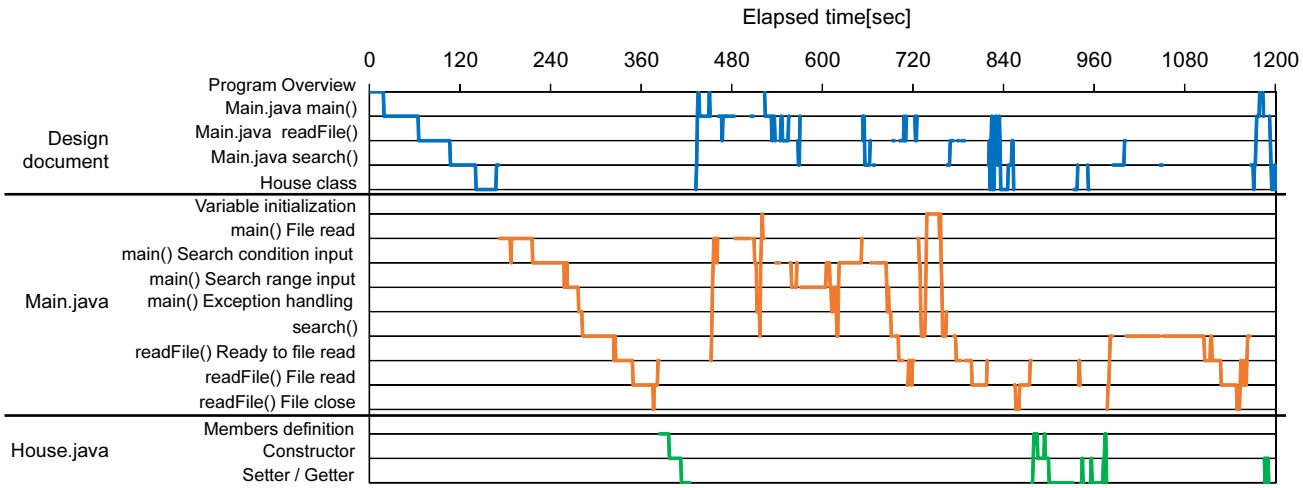

図 5.1　指示有りグループの視線移動

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 ($p = 0.335$).

図 5.3 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 16.8% の差が見られた．また，この時点で指示有りと指示無しのバグ発見率の差に対して Welch の t 検定（両側検定）を行った結果，$p < 0.05$ で有意な差が見られた．バグ発見率の差は作業開始 30 分経過時点で最大値 17.1% となったが，この時点での有意差は見られなかった ($p = 0.073$).

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

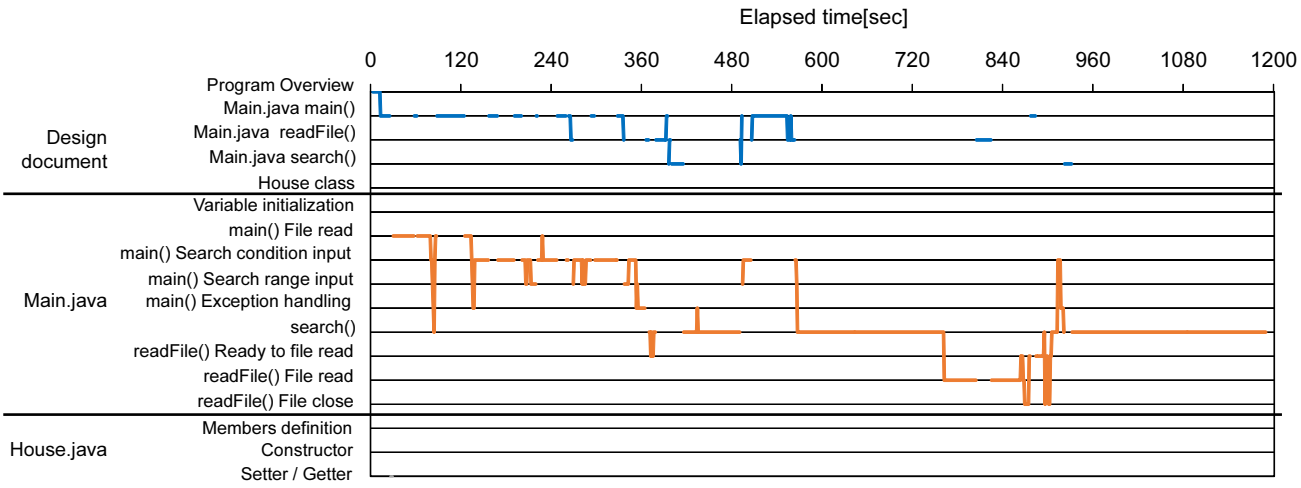

図 5.2　指示無しグループの視線移動

の半分とも言われており[*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という結果には，時間をかけたレビューにおいて効果を出していることから，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．

5.2.1　指摘精度

図 5.4 に各グループのバグ指摘回数を，図 5.5 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 12.1%，21 分から 30 分で 54.3% 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，バグ発見率の差が優位に開いた作業開始 26 分経過時点にグループ間の指摘精度の差が最も大きくなる時間帯が一致しており，指示有りグループは指摘精度が向上したため，バグ発見率が高くなったと言える．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

[*1] http://msdn.microsoft.com/ja-jp/library/ms182019.aspx

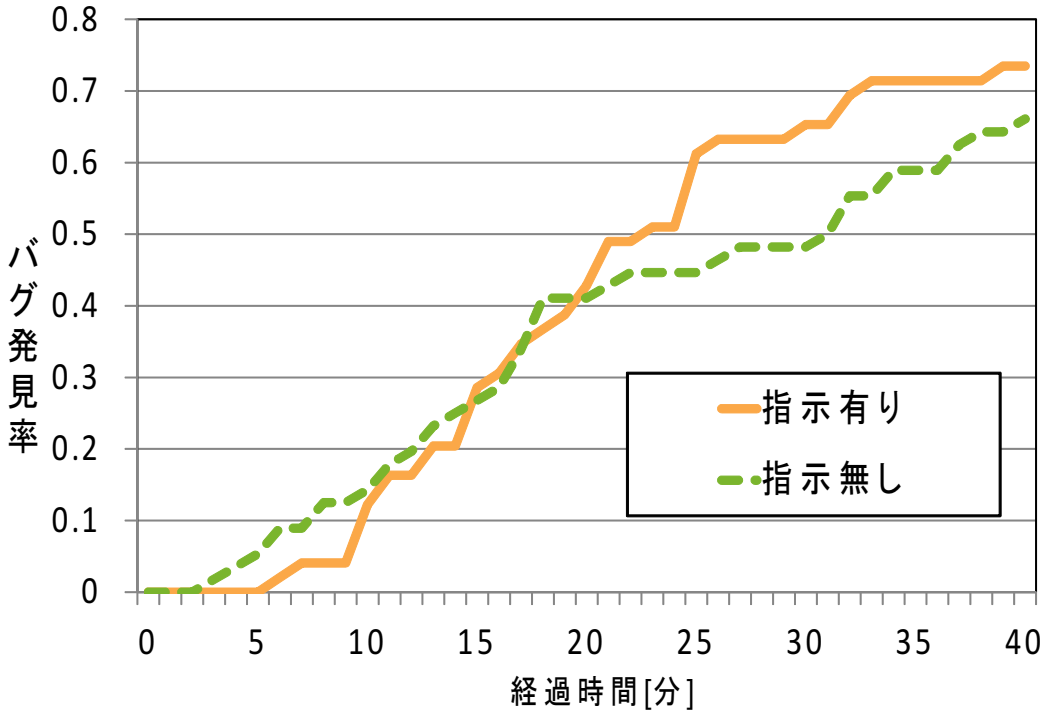

図 5.3　バグ発見率の推移

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．また，図 5.3 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 66.1% であったのに対して，指示有りグループは 32 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，指示有りグループの被験者が 32 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多く行えず，有効な指摘も少なくな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．

5.3　Prefix span 法による頻出パターンの抽出

指示が被験者の提示物の読み方に与える影響を調べるため，被験者実験で得られたレビュアーの視線情報をもとに，実験中の被験者の視線情報を分析した．分析には配列データベースから頻出パターンを抽出するアルゴリズムである Prefix span 法 [14] を用いた．Prefix span 法を用いることで，グループごとに被験者の提示物の読み方の傾向を抽出

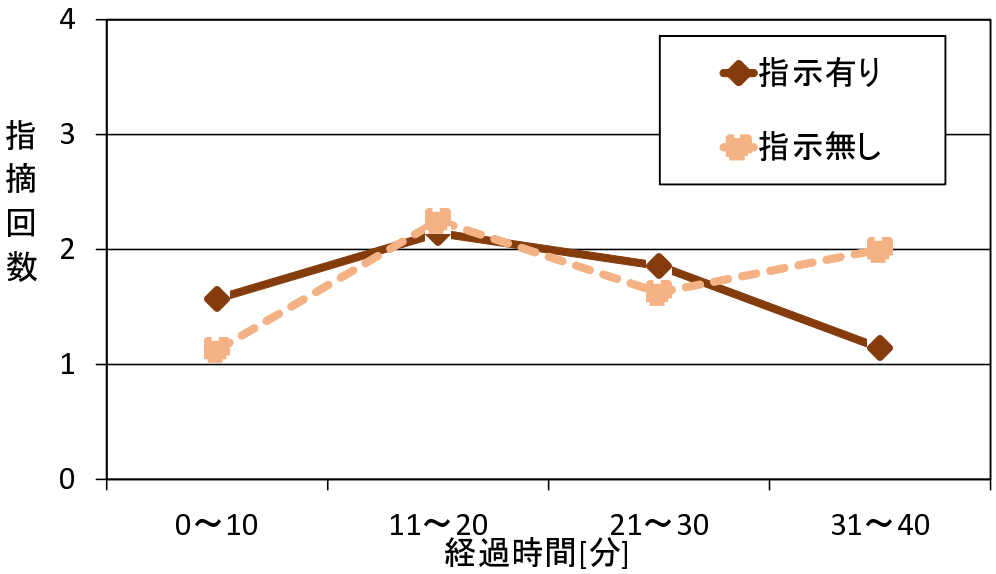

図 5.4　バグの指摘回数

することができる．分析では，Prefix span 法を用い，被験者の視線情報から指示有りグループの被験者のみが行った読み方のパターンを抽出し，指示による読み方の差を調査した．分析において，提示物はブロックに分割し，Prefix span アルゴリズムを実装したプログラムでパターンを抽出した．抽出するパターンはブロックの連続数（パターン長）の最小値を 4，最大値を 15 とした．また，パターン間にブロックの読み中や読むブロックの違いがあっても違いが 2 つまでなら誤差として許容し，指示有りグループの被験者のみが行ったパターンを抽出した．図 5.6 に示す本実験に使用したプログラムは，物件情報を格納したデータファイルから，坪数・駅からの距離・家賃を検索条件とし，条件に一致する物件を検索・表示するプログラムである．図内の番号はメソッドの呼び出し順序を表す．はじめに main メソッドが readBukkenFile メソッドを呼び出し，データファイルから取得した物件データを配列 blist へ格納する．次に，main メソッドが検索条件を取得し，bukkenSearch メソッドを用いて検索条件に適合する物件を探索する．bukkenSearch メソッドは blist 内に格納されたデータから，getter を用いて坪数・駅からの距離・家賃を取得し，条件に適合するか調べる．検索結果は bukkenSearch メソッド内で showHouse メソッドを呼び出すことで表示する．

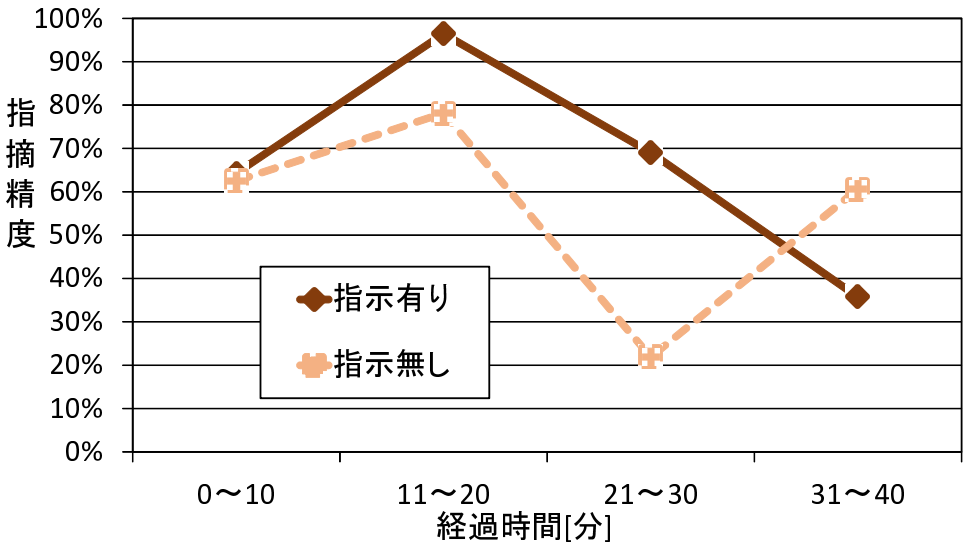

図 5.5　バグの指摘精度

図 5.7，図 5.8 に指示有りグループの被験者に見られた提示物の読み方の一例を示す．図 5.7 は getter や showHouse メソッドを確認した後，showHouse メソッドで使用されている blist・ MAXHOUSE を定義している readBukkenFile メソッドの設計を読み，次に bukkenSearch メソッドでのそれらの使用状況を確認し，最後に main メソッド内で readBukkenFile メソッドが使用されていることを確認している．

この読み方は，途中に変数の定義を確認するために設計書の確認を挟み，図 5.6 の 3,4 の矢印から 2 にかけてのプログラムの流れを遡っていると言える．図 5.8 は指示有りグループの被験者が bukkenSearch 上で，変数 MAXHOUSE，物件リスト blist，ゲッターが使用されていることに気づき，MAXHOUSE や blist が，設計書でどのように定義されているか確認し，再度 bukkenSearch でのそれらの使用状況を照らし合わせ，変数 MAXHOUSE の初期化がどのように行われているか確認したことを表している．この読み方はプログラムの流れを確認する読み方ではなく，変数が定義通り初期化・使用されているか確認するための読み方であると言える．

これらの結果から，読み方の指示は変数やメソッドが設計書で指定された通り使用されていることの確認や，変数やメソッドが実際に使用されているソースコード上の場所の確認をプログラムの流れに沿って行わせることでプログラム理解を促進させているとわか

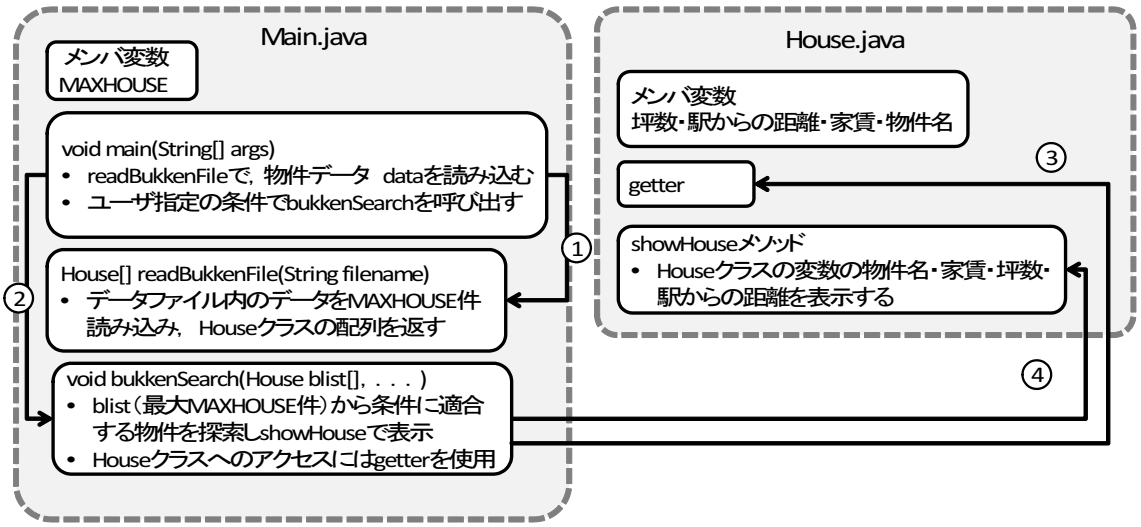

図 5.6　プログラムの概要

る．よって，指示をすることで，被験者のプログラム理解が効果的に行われたと言える．本研究から，レビュー開始時に見るべき点を指示することがプログラムの流れ確認作業を促し，バグの指摘精度が向上し，結果，バグ発見率が向上することがわかった．また，現状の指示内容では設計書の精読にかける時間に，指示有りグループの被験者間も差が出て，プログラムをレビュー開始時に理解することの重要性が伝えきれていないとも考えられる．よって今後の発展としては手順を以下のように変更し，プログラムへの理解をより向上させることも有効であると考えられる．

1. 設計書をよく読み，理解してください．
2. コード全体に通し，プログラムの全体像を把握してください．
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，プログラムの流れを理解してください．

この指示を用いることで，より具体的にレビュアーへプログラム理解のための読み方を伝えられると考える．

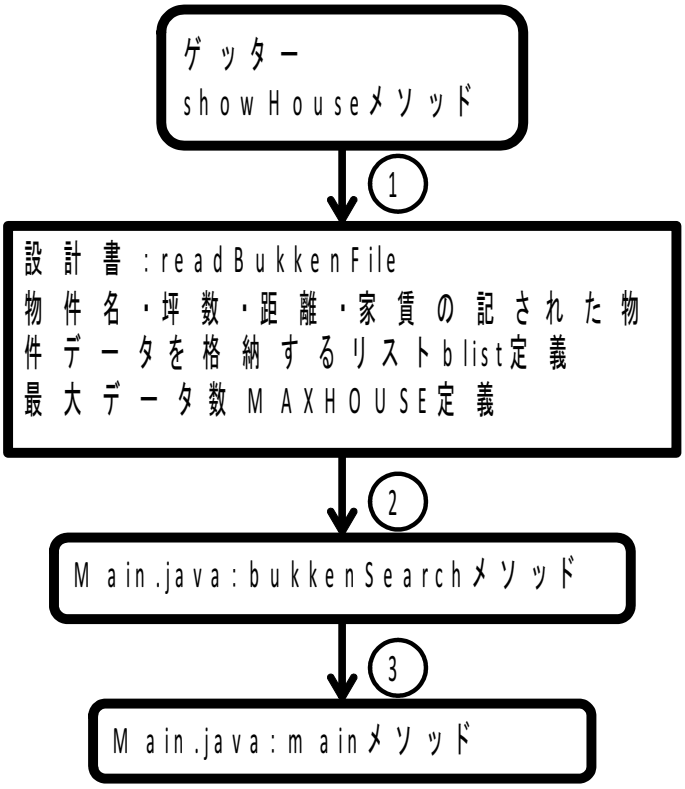

図 5.7　読み方パターンの例 1

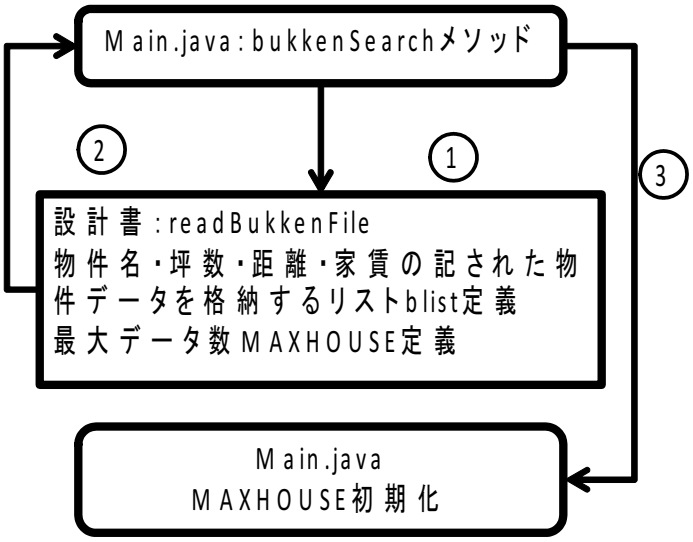

図 5.8　読み方パターンの例 2

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．本稿で提案した読み方はレビュアーによって読み方に個人差が生じない内容を用いている点で，従来手法に比べ具体的であると言える．

実験の結果，レビュー終了時点において，指示有りグループのバグ発見率が指示無しグループに比べて 7.4% 高くなった．レビュー開始後 26 分の時点で，指示有りグループのバグ発見率が指示なしグループを 16.8% 上回り，この差は有意であることを確認した．また，両グループの指摘回数に大きな差が見られなかった一方，指摘精度については指示有りグループが指示無しに比べ最大 54.3% 高かった．この結果から，指示有りグループの被験者は読み方の指示によって，バグの指摘回数が増えたのではなく，指摘そのものの精度が向上し，正確にバグを発見できるようになり，バグ発見率が向上したと言える．また，Prefix span 法による被験者の視線移動分析では，本校で採用した読み方の指示は，プログラムの流れに沿って変数やメソッドの位置や設計書との整合性の確認を行わせ，プログラム理解を促進させていることがわかった．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，被験者へプログラム理解を促進させ，レビュー効率の向上につながったと言える．

また，提案した指示は CBR で使用されるチェックリストの先頭に記述して利用可能と考えられる．現在の提案では，チェックリストと別に作業者は指示内容についても注意する必要があるため，指示内容をチェックリスト内でまとめるほうがより現場で使いやすくなると言える．さらに，指示内容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，将来的には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，初学者によるレビューにおいて効果が期待できる．

謝辞

はじめに，指導教官として研究に多大なるご指導をいただいた上野先生に厚く御礼申し上げます．また，研究用のデータ取得のため，実験に快く協力してくださった被験者の方々に感謝いたします．

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．

参考文献

[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,” PhD 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] 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).

[5] F. Lanubile, G. Visaggio：“Evaluating Defect Detection Techniques for Software Requirements Inspections,” ISERN pp. 00-08(2000).

[6] 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).

[7] 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).

[8] 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).

[9] S. Zhai, C. Morimoto, and S. Ihde：“Manual and gaze input cascaded (MAGIC) pointing,” In Proceedings of The SIGCHI Conference on Human Factors In Com-

puting Systems ‘ 99, pp. 246-253(1999).

[10] 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).

[11] 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).

[12] Hidetake Uwano：“Measuring and Characterizing Eye Movements for Performance Evaluation of Software Review”, Ph.D. thesis, Nara Institute of Science and Technology, (2009).

[13] Boris Beizer：“ソフトウェアテスト技法”(1994).

[14] 塔野薫隆, 北上始, 田村慶一, 森康真, 黒木進：“Modified PrefixSpan 法を用いた頻出正規パターンの抽出をめざして”, 日本データベース学会 Letters,Vol3, No.1, pp. 61-64, (2004).