Vol.2019-SE-203 No.9
2019/12/12

組込みシステム開発のデバッグにおける視線と熟練度の関係分析

馬場 建[1,a)]　槇原 絵里奈[1,b)]　米田 浩崇[2,c)]

概要：組込みシステム開発はソフトウェア開発と異なり，ハードウェアが密接に関わる．組込みシステムでは考慮すべき要素が増えるため，初学者はシステムが動かない原因の箇所を推定し，修正することは難しく，初学者のデバッグ効率の向上が課題となる．デバッグ効率を向上させる方法の一つに，熟練者のデバッグのコツを初学者に教示する方法が考えらえる．組込みシステム開発のデバッグ作業の視線において，熟練度の差がどのように現れるかを被験者 7 名の被験者実験により調査した．調査の結果，実験時間全体をまとめて分析した場合，時間分布や注目物体の遷移確率は熟練者と初学者の間で有意な差がなかった．注目物体の時間的推移について調査した結果，熟練度によって時間ごとの注目物体の傾向に違いがあった．傾向の違いを明らかにするために，時系列データを時間分割して注目物体の遷移確率を調べたところ，熟練者は終盤でソースコードと回路の双方に誤りが存在する可能性を考慮しながらデバッグするといった傾向が見られた．このように注目物体の時間的な傾向を基に，初学者に対してシステムの修正方針を教示できると考える．

1. はじめに

IT 人材の不足は深刻化しており，2030 年には最大で約 79 万人の不足が起こると予想されている [1]．IT 人材の中でも，IoT 市場規模の拡大に伴い，IoT 技術者の需要が増加すると予想されている [2]．IoT の開発は，組込みシステム開発を基礎技術として，ネットワークやセンシング，データの分散処理などの応用技術が必要であり，技術の習得が容易でない．IoT の基礎技術である組込みシステム開発の効率化は，今後のソフトウェア工学の課題になると考えられる．

組込みシステム開発は，一般的なソフトウェア（以降，SW）開発に加えてハードウェア（以降，HW）の要素が増える．組込みシステム開発における，システムが正常に動かない原因，すなわち誤りの箇所は，SW だけでなくマイコンや回路などの HW にも発生しうる．多くの場合，HW と SW の誤りは見分けがつかず，システムの誤り箇所を適切に推定し，修正することは初学者にとって容易でない．

組込みシステム開発において，初学者の教育を支援するために，学習キットの作成に関する研究が行われている [3][4]．しかし，組込みシステム開発のデバッグは多くの要素を考慮しながら行う必要があるため，誤り箇所を推定するために熟練度によって差が生じやすく，その差を埋めるアプローチでの教材が必要だと考えられる．熟練者の持つ組込みシステム開発のデバッグ特有のコツや，初学者のつまずきやすい箇所を分析することで，デバッグ効率向上のための教材の拡充が可能である．そこで本研究では，組込みシステム開発における熟練度ごとの特徴を明らかにするために，アイトラッカーを用いた視線分析を行う．具体的には，組込みシステム開発のデバッグ作業における視線動作を定量的に分析し，熟練者と初学者のデバッグ過程の違いを明らかにすることを目的とする．視線の特徴として，作業者の無意識的な能力が動作に反映されるため，先行研究では視線情報を用いた熟練度の差に関する研究が行われている．視線情報を基に熟練者のコツを明らかにできれば，初学者にコツを教示することでデバッグ効率の向上に貢献できると考える．

本論文の構成は以下の通りである．2 章では組込みシステムと熟練度の分析方法についての前提知識や関連研究について述べる．3 章では本研究で行った被験者実験について実験条件や実験環境などについて述べ，4 章では実験結果の考察について述べる．5 章では結論と今後の展望について述べる．

[1] 同志社大学理工学部インテリジェント情報工学科
京都府京田辺市多々羅都谷 1–3
Tatara Miyakodani 1–3, Kyotanabe, Kyoto 610–0394, Japan
[2] 同志社大学大学院理工学研究科情報工学専攻
[a)] tbaba@mikilab.doshisha.ac.jp
[b)] emakihar@mail.doshisha.ac.jp
[c)] hyoneda@mikilab.doshisha.ac.jp

2. 準備

2.1 組込みシステム開発のデバッグに関する課題

組込みシステムの開発は，一般的な SW 開発に加えて HW の要素が増える．HW の誤りとして想定されるものの一例を，各要素ごとに下記に挙げる．

- 書込み：ケーブル断線，ポート無認識，モード不一致
- 電源：短絡，電圧不一致，電源不足，GND 非共通
- マイコン：接続ピン誤り，仕様の無理解，実行速度
- 回路：故障，配線誤り，定格違反，経年劣化

多くの誤りが HW に起こり得るが，その誤り箇所を推定することは容易でない．誤り箇所の推定が難しい理由として，HW はエラーの出力をすることが手軽でないことが考えられる．SW はビルド時や実行時にエラーを文字出力することができ，システムの状態を詳細に把握することができる．しかし，HW は回路上の数少ない入出力装置の挙動からシステムの正常性を確認する方法か，テスターやオシロスコープといった計測器を用いて一箇所ずつ調べる方法が一般的であり，システムの状態を詳細に把握することが容易でない．HW の誤りが原因でシステムが正常に動作しない場合でも，その誤りの箇所を把握していなければ，SW の実装ミスと判断する可能性がある．SW の誤りが原因でも，同様の判断ミスが起こる可能性がある．HW と SW のどちらに誤りが生じているかの推定には，経験から獲得できるコツが関係しており，初学者にとってデバッグが難しい理由の一つであるといえる．

2.2 視線計測による能力評価

2.2.1 視線の特徴と分類

視線には，思考プロセスや能力によって動きが変化するという特徴がある [5]．視線の動きには，作業者が意識的に行っている視線移動に加えて，作業者が言語化できないような，無意識的な判断による視線移動が反映される [6]．したがって，熟練者のもつ無意識的で伝達不可能だったコツを，視線分析から明らかにできる可能性がある．

人間の視線は主に二つの動きに分類できる [7]．一つ目は saccade と呼ばれる視線の高速移動で，注目物体の移動速度が 30 deg/s より高速のときや，離れた位置の物体を見る際に起こる．二つ目は，fixation と呼ばれる視線の停留で，注目物体の移動速度が 30 deg/s より低速のときや，静止物体を見る際に起こる．saccade と fixation は交互に行われ，視覚認知の多くが fixation 中に行われている．

2.2.2 アイトラッカーの特徴

アイトラッカーとは，人間の視線を定量的に取得できるデバイスのことである．アイトラッカーは非接触型とウェアラブル型の 2 種類に分類できる．

非接触アイトラッカーは，PC 作業と同時に使用し，PC のディスプレイに対する視線のみを取得することができる．ウェアラブル型と比較して安価であり，ディスプレイに対する視線を高精度に取得できる．ディスプレイの座標と視線の fixation の座標を一致させて取得できることから，注目物体が何であるかという定量分析が容易であり，ソースコードリーディングや，パッケージや広告などのデザイン，web 閲覧解析などに利用されている [8][9][10][11].

ウェアラブルアイトラッカーは，一人称の視線映像上のあらゆる物体に対する視線を取得することができる．非接触型と比較して高価であり，装着者が見たあらゆる対象への視線を取得できるが，文章を読むというような高精度な視線の取得ができない．装着者が実際に見た視線映像上の fixation の座標が取得できることから，視線映像上の物体の位置が常に変化し続けるため，注目物体の認識を定量的に行うことが容易でない．そのため，ウェアラブルアイトラッカーを使用するほとんどの研究において，注目物体の判断は人間の手作業で行われている．

2.2.3 アイトラッカーによる定量分析

アイトラッカーを用いることで，視線の fixation の座標を取得することができる．fixation の座標から単純に分析できる定量データは，注目座標の分布や停留時間，移動距離や移動方向などがある．fixation の座標に関する分析手法は，統計的分析，時系列分析，認知プロセスの推定の 3 種類ある [5]．統計的分析は注目物体に関する時間的な分布を，時系列分析は注目の順序関係を調べる方法である．そして認知プロセスの推定は，PC の作業ログと視線の遷移を関連付けることで，人の認識過程を推定する方法である．

fixation の座標と映像上の物体領域情報を組み合わせると，視線情報から分析できる事項が多くなる．例えば，ソースコードなどの文章を分析したい場合，文章の行の幅や画面のスクロールで変化するピクセルの量が把握できれば，作業者が読んでいる位置を機械的に判別することができ，それを実現するプラットフォームも存在している [12]．本の種類や検品作業など，注目する物体ごとに分析を行う場合，物体位置の領域を機械的に分析することは難しいが，定量的に判別する研究も行なわれている．江川らは，視覚情報の持つ生来的な性質を利用したセグメンテーションに基づく領域分割を行なった [13]．志賀らは，作業者が読んでいる書類を自動判別するシステムを作成した [14].

2.2.4 視線情報のモデル化

視線には無意識的な暗黙知が反映されるため，視線分析により暗黙知が定量的に表せるとされている [6]．Paul は，ソースコードリーディングにおけるコツを視線から分析し，ソースコードの要素の意味に関する遷移を隠れマルコフモデルとして表した [15].

Busjahn らは，ソースコードリーディングにおいて注目した関数の遷移に関してマルコフモデルを生成した [16]．花房らは，プログラミング中の視線の停留点の分布と成績情報を基に，Support Vector Machine を用いて学習者のレ

ベル分けを行なった [17]．このように，視線動作をモデル化することで，熟練度による動作の違いを明らかにし，被験者の差を算出することが可能となる．

2.3 視線から明らかになる熟練者のコツを教示する効果

視線の分析によって得られたコツを作業者に教示することで，作業効率が上がることが知られている．應治らは，ソースコードレビューにおける熟練度の差を視線動作から計測し，設計書やソースコードの読み方を作業者に教示することでレビュー効率が変化するかを検証した [8]．実験の結果，教示ありのグループは誤りの発見速度が上昇し，誤りの発見率も向上したと報告している．

他のソフトウェア工学の研究として，プログラミング実装課題に関して，高得点群のソースコード上の fixation に関するヒートマップを初学者に提示することで，低得点群の回答速度が向上することが報告されている [18]．ソフトウェア工学以外の看護やドラムの演奏などの分野でも，熟練者のコツを教示する効果が報告されており，熟練度に関する視線の分析は広い分野で作業効率の向上に貢献している [19][20]．

3. 組込みシステム開発に関する視線分析実験

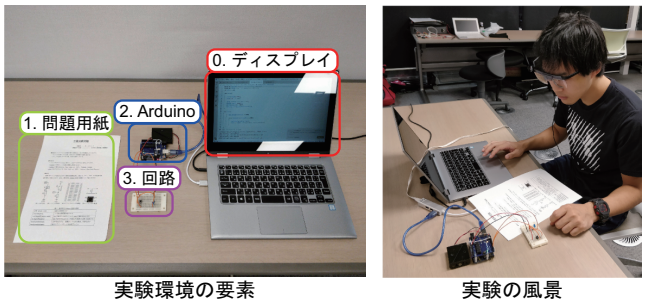
3.1 実験目的

2.2.4 項で述べたように，視線情報のモデル化で熟練者のコツを明らかにでき，2.3 節で述べたように，熟練者のコツを初学者に教示することで作業効率が向上できるということが，組込みシステム開発のデバッグ効率の向上にも応用できると考えられる．したがって本研究では，組込みシステム開発のデバッグ作業における視線動作を定量的に分析し，熟練者と初学者のデバッグ過程の違いを明らかにすることを目的とする．そして得られた熟練度の差は，初学者のデバッグ効率を向上するための，組込みシステム開発のコツとして教材化できると考える．したがって，組込みシステム開発のデバッグ中の視線情報から数学的モデルを生成することで，熟練度の差を生む要因を明らかにするための被験者実験を実施した．熟練度の違いによって生まれる差を分析するために，被験者に組込みシステムのデバッグを行う課題を与え，デバッグ中の視線情報と操作ログを記録する．熟練者と初学者の視線動作の違いは，経験を積んで熟練することで得られた組込みシステム開発のデバッグ特有の技能であり，本研究ではこの違いを熟練者の持つコツとして考える．

なお，本研究は組込みシステム開発に固有の特徴を分析することが目的であるため，詳細な修正内容に関して，回路とソースコードを個別に分析することは行わない．

3.2 実験条件

被験者は，組込みシステム開発経験がある大学生の 7 名



実験環境の要素　　実験の風景

図 1 実験環境の要素と実験の風景

である．本研究では経験により得られる組込みシステム開発に固有の特徴を明らかにしたいため，経験年月数が長い被験者を熟練者と定義する．したがって，組込みシステム開発経験が 1 年以上である 3 名を「熟練者」，1 年未満である 4 名を「初学者」と定義する．初学者の中に未経験者は含まれていない．

熟練者 1 名は開発経験が 3 年であり，残りの 2 名は開発経験が 1 年半であった．すべての熟練者は，使用経験のあるマイコンが 4 種類以上であった．また，サークルやアルバイトで他人に教育をした経験があるなど，日常的に組込みシステム開発を経験している．初学者 3 名は開発経験が 1 か月であり，残りの 1 名は開発経験が 6 か月であった．すべての初学者は，使用経験のあるマイコンが 1 種類であった．また，回路図を見て回路を組んだ経験がない初学者が 2 名いた．組込みシステム開発に関わる機会は，サークルやハッカソン，研究活動，実験の授業と多岐に渡った．

被験者に与える課題は，スイッチの押下時に一定時間間隔で LED が点滅し，押下時以外は LED が消灯するシステムの構築である．被験者がデバッグを開始してから，システムのすべての誤りを修正するまでの期間における視線情報と操作ログを取得する．

3.3 実験環境

3.3.1 被験者に与えた開発環境

実験は**図 1** に示す環境で実施した．環境の構築に用いた組込みシステムの要素を下記に述べる．

- ディスプレイ：ソースコードの閲覧および編集と，コンパイル結果などの文字出力
- 問題用紙：システムの完成形の説明や，システムで実装する回路図，ソースコードで使用する関数，ソースコードのコンパイル方法などを記述
- Arduino：回路の電圧を制御するマイコン
- 回路：デジタル入出力に関する回路．LED とスイッチと抵抗およびジャンパー線を用いた実装

3.3.2 課題のシステムの初期状態

被験者に課題として与えるシステムには，ディスプレイと回路のそれぞれに二つずつ誤りが含まれている．システムの完成目標の回路図と，誤りが含まれた回路図を**図 2** に

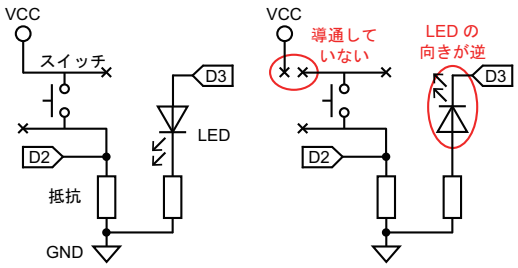


図 2　正解の回路（左）と誤りがある回路（右）

図 3　Tobii Pro Glasses2

示す．ソースコード側には，動作の待ち時間に関する誤りと，回路の状態を正しく反映できるかを問う誤りがある．回路側には，接続されているか確認が必要である誤りと，電子部品の特性に関する知識が必要な誤りがある．これらの誤りは，著者が三年間のロボット研究会の活動や教育の経験から，ディジタル入出力の回路に関してよく発生する誤りを選択したものである．

3.3.3　実験中に取得する情報

被験者がシステムの誤りをデバッグをする間，被験者の視線情報と PC の操作ログを取得した．視線情報を取得するために，**図 3** に示す Tobii Pro Glasses2 というウェアラブルアイトラッカーを使用した[*1]．視線情報取得のサンプリングレートは 100 Hz であり，視線映像のフレームレートは 25 fps で解像度は 1920 × 1080 px である．視線が fixation の状態と判定する条件は，60 ms の間視線の移動距離が 30 deg/s 以下だった場合である．

PC の操作ログからは，コンパイル結果とソースコードの編集履歴を取得した．操作ログを取得する環境を構築するために，Visual Studio Code を開発環境として用意した．「Arduino」という拡張機能[*2]を Visual Studio Code にインストールし，その拡張機能のソースコードを変更してコンパイルが実行されるたびにテキストファイルに追記するように実装した．ソースコードの編集履歴を保存するために，ソースコードの保存時にファイルを別名保存できる「Local History」という拡張機能[*3]をインストールした．Visual Studio Code の「Auto Save」機能を用いて，最後にソースコードの変更が発生してから 1 秒後にファイルを自動保存するようにし，自動的にソースコードの編集履歴を別名で書き出せるようにした．

3.3.4　アイトラッカーによる物体検出

2.2.3 項で述べたように，ウェアラブルアイトラッカーを用いた注目物体検出は，人間が手作業で行うことが主流であった．本研究では，注目物体を調べる過程をコンピュータに処理させるため，YOLO v3 という手法で物体検出器を作成し，注目物体の判別を自動的に行えるようにした [21]．注目物体として認識するオブジェクトは，3.3.1 項で示した 4 要素のみに制限する．注目物体の自動化はあくまで実験結果の解析を補助する目的であり，正しく認識できているかを人の目で最終確認を行う．

注目物体検出のために，予備実験時の被験者の視線映像を用いて，物体検出器を作成した．物体検出器と視線座標を用いた注目物体検出の処理は，下記のような手順で行なった．

手順 1　視線情報と視線映像を入力
手順 2　視線の fixation が生じた映像のフレームに移動
手順 3　物体検出器にそのフレームの画像を入力
手順 4　物体検出器から物体位置を長方形領域として取得
手順 5　fixation の座標が物体領域内かを判断
手順 6　fixation の座標が物体領域内になければ，視線が注目物体の遷移過程であると捉えて一つ前の注目物体を現在の注目物体と認識
手順 7　fixation の座標が物体領域内にあれば，物体領域の重心までの距離の比が小さいものを現在の注目物体として認識

3.4　実験手順

被験者実験は，下記に示す手順で実施した．

手順 1　誤りがあるシステムを被験者に与える
手順 2　被験者に対してシステムの完成形を見せながら課題の目標を説明
手順 3　被験者に対して課題内容やコンパイル方法，実験中の注意などの説明を行う．また次の条件にあてはまる被験者に対しては，問題用紙に書かれている内容を見せながら，関数や回路図を簡潔に説明した
　条件 1　Arduino 言語の使用経験がない
　条件 2　回路図からの回路の実装経験がなく，実体配線図や写真からのみ回路の実装経験がある
手順 4　被験者はシステムのデバッグを行う課題を遂行
手順 5　組込みシステム開発における，経験年月数や開発に携わる機会，使用経験があるマイコンの種類などの項目に関するアンケートを実施

[*1] https://www.tobiipro.com/ja/product-listing/tobii-pro-glasses-2
[*2] https://marketplace.visualstudio.com/items?itemName=vsciot-vscode.vscode-arduino
[*3] https://marketplace.visualstudio.com/items?itemName=xyz.local-history

表 1　平均回答時間と物体の平均注目時間の割合

	時間 [分]	p0	p1	p2	p3
熟練者	14	0.63	0.18	0.03	0.17
初学者	32	0.59	0.20	0.05	0.16

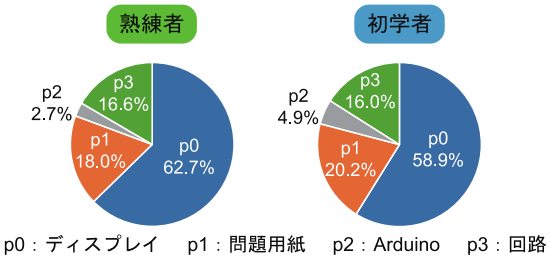


図 4　実験時間全体の注目物体の時間分布

3.5　実験結果

3.5.1　実験時間の全体的な傾向に着目した分析

被験者実験で取得した情報から，熟練度の差を起こす要因を分析した．熟練者と初学者の修正にかかった平均時間と，各物体を注目していた時間の割合を表 1 に，時間分布の円グラフを図 4 に示す．

p0〜p3 は，それぞれディスプレイ，問題用紙，Arduino，回路のいずれかに被験者が注目している状態である．熟練者は初学者と比較して短時間で誤りを修正しており，熟練度がデバッグにかかる時間に影響していることがわかる．物体の注目時間の割合は被験者によってばらつきが大きく，有意水準 5%の welch の t 検定で有意な差は見られなかった．熟練度に関わらずディスプレイに注目している時間分布が大きく，次に問題用紙，回路，Arduino という順番で注目分布が小さくなることが共通していた．

次に，実験開始から終了までの視線の停留ごとの注目物体の推移を調べた．熟練者 3 名の注目物体の推移を図 5 に，初学者 4 名の注目物体の遷移を図 6 に示す． 熟練者は序盤に回路を中心に注目する傾向があり，次にディスプレイ，最後に回路に注目する順番でデバッグを進める傾向がある．初学者は序盤に回路とディスプレイ間の注目の遷移を繰り返す傾向がある．

熟練者と初学者の間で，注目する箇所の偏りがあるように見られるため，注目物体の遷移確率で熟練度の差が生じるかを調査した．熟練者と初学者の注目物体の遷移確率に関するマルコフモデルを図 7 に示す．連続して同じ物体に遷移する確率は非常に高いため，本グラフでは除いている．図の矢印の太さは，遷移確率の高さを視覚的に表したものであり，矢印が太いほど確率が高い．有意水準 5%の welch の t 検定の結果，すべての遷移確率において有意な差はなかった．すなわち，熟練度に関わらず，問題用紙を中心とした遷移確率が高いことが本実験で観測された．

3.5.2　実験時間の部分的な傾向に着目した分析

3.5.1 項では，実験時間全体をまとめて分析したが，熟練度の違いによる有意な差が見られなかった．しかし，図 5

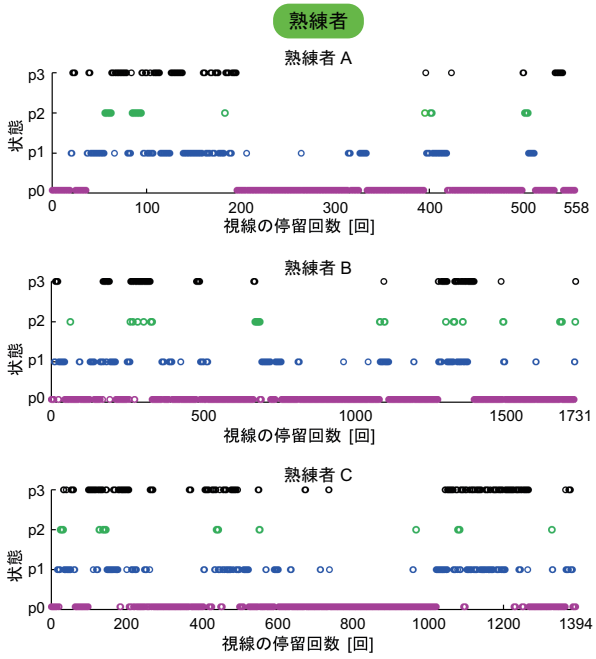


図 5　熟練者 3 名の注目物体の時間的推移

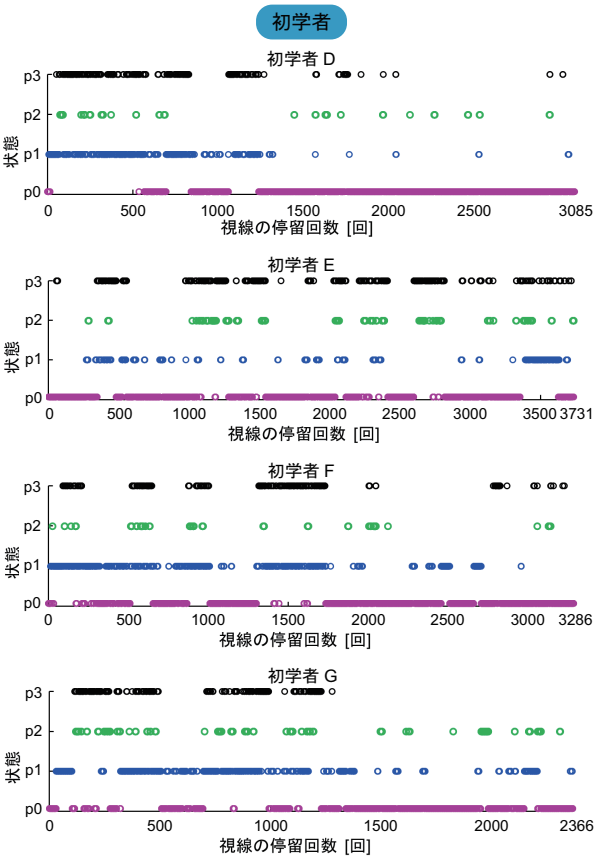


図 6　初学者 4 名の注目物体の時間的推移

と図 6 から注目物体の時間的推移において，熟練度によってデバッグの方針が違うことが予想されたため，実験時間を分割して視線動作の違いが現れるかを調査した．そこで，熟練者がどのような方針でデバッグを行うかを調べるために，実験結果を fixation の回数で三等分に分割して分析す

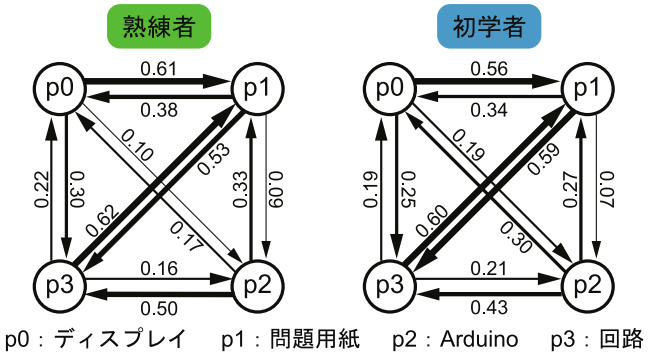


図 7　実験全体における注目物体のマルコフモデル

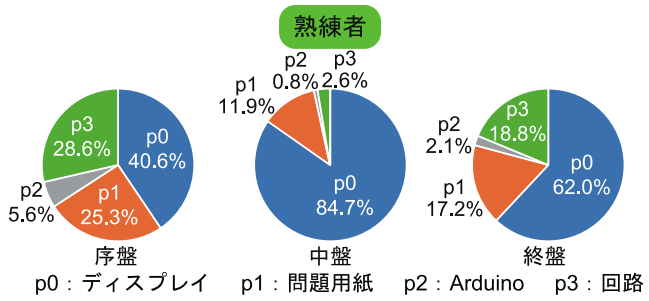


図 8　時間分割したときの熟練者の注目物体の時間分布

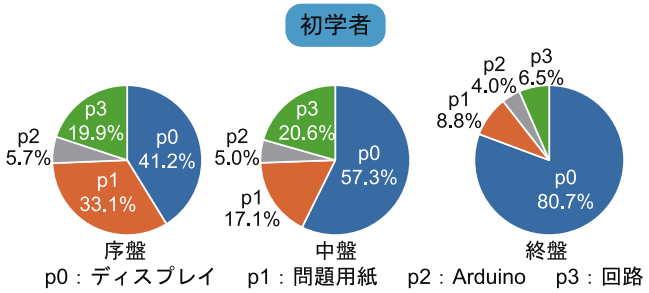


図 9　時間分割したときの初学者の注目物体の時間分布

ることで，時系列的なデバッグ箇所の違いがあるか検証した．実験時間を三分割して出力した注目時間の分布のグラフのうち，熟練者の時間分布を図 8 に，初学者の時間分布を図 9 に示す．熟練者と初学者の時間分布を三つの段階で比較すると，中盤の時間分布に顕著な違いがみられた．

実験結果を分割することで時間分布に違いが現れたため，各段階における注目物体の遷移確率をモデル化し，傾向が異なるかを詳細に調べた．熟練者と初学者の実験の序盤における注目物体のマルコフモデルを図 10 に，中盤における注目物体のマルコフモデルを図 11 に，終盤における注目物体のマルコフモデルを図 12 に示す．

被験者実験の開始から終了までを三分割にして注目物体の遷移確率を分析した場合，各遷移確率に関して有意水準 5%の welch の t 検定をした結果，中盤と終盤において，図 11，図 12 の確率値が太字で示されている遷移に関して有意差があった．

序盤において，熟練度に関わらず，ディスプレイや回路から問題用紙に向かう遷移確率が高かった．中盤において，熟練者は問題用紙からディスプレイへの遷移確率が，問題用紙から回路への遷移確率と比較して高かった．初学者は Arduino から回路への遷移確率が高く，問題用紙からディスプレイと回路に向かう遷移確率は近い値を示した．終盤において，熟練者はディスプレイと回路間の遷移確率の値が近かった．また問題用紙からの遷移確率は，ディスプレイと回路の両方の遷移確率が近い値を示した．終盤において初学者は，回路からディスプレイに向かう遷移確率が高い値だったが，逆向きの遷移確率は低い値だった．また問題用紙からディスプレイへの遷移確率は高い値だったが，問題用紙から回路への遷移確率は低い値だった．

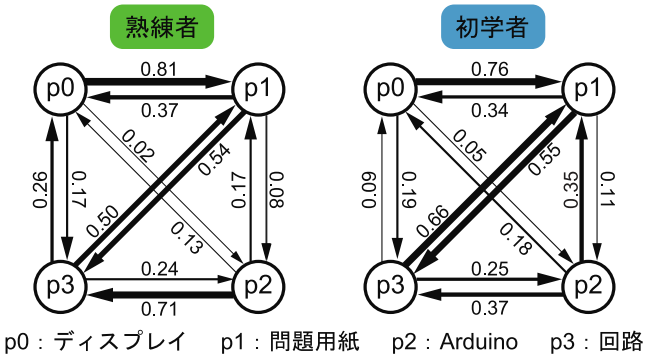


図 10　実験の序盤における注目物体のマルコフモデル

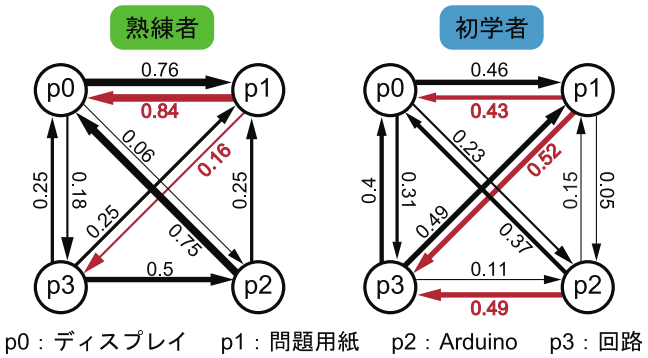


図 11　実験の中盤における注目物体のマルコフモデル

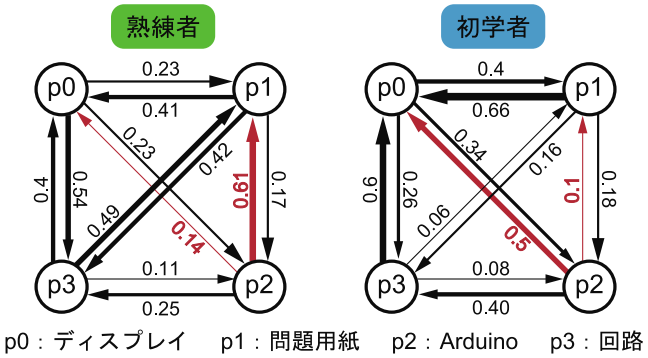


図 12　実験の終盤における注目物体のマルコフモデル

4. 考察

実験開始から終了までの時間分布が熟練者と初学者で有意な差がなかったのは，熟練度に関係なく，実験に用いた

各要素が必要である時間に違いがないためであると考えられる．例えば，ディスプレイへの注目時間が最も長かったのは，ソースコードのタイピングやコンパイルに時間がかかることが理由として考えられる．また今回の実験で，システムの誤りが存在しなかった Arduino は，最も注目時間が短くなった．しかし，被験者の一部に対してヒアリングを行い，デバッグ時に使用する要素に関して優先順序をつけてもらった結果，最も問題用紙を重視し，次に回路，ソースコード，Arduino という優先順序の回答や，最も回路を重視し，次にソースコード，問題用紙，Arudino という優先順序の回答が得られた．実装にかかる時間が違うため，デバッグにかかる時間分布の降順は，被験者が回路を重要視するという主観的な優先順序と一致しなかった．したがって，時間分布から各要素の重要度を判別することはできなかった．

実験全体におけるマルコフモデルが，熟練度によって有意な差がなかったことから，各要素をどのように関係付けているかについて違いがないことがわかる．例えば，被験者全員が問題用紙への参考を中心として，回路やソースコードのデバッグ作業を行なっているということがわかる．

熟練度によって注目物体の時間的推移の傾向が異なったのは，デバッグの修正方針の違いが関係していると推測される．熟練者は実験全体においてはじめに回路を修正するという方針を持っていたことが示唆される．熟練者がはじめに回路を修正したのは，システムの挙動を正しく見るために確実な修正を試みたという理由が考えられる．実際に熟練者の一部へシステムの修正方針についてヒアリングを行ったところ，「まず，回路を見る．回路が間違っているとプログラムを書き込んで試しながらデバッグできない.」という回答や「試しにシステムを実行したとしても，そもそも何も動作しなかったら，プログラムの修正をしようがないため，最初に回路を見た.」という回答があった．実験結果の分析とヒアリングの結果から，2.1 節で述べたように，HW 側でシステムの状態を把握することが容易でないという性質があることから，先に回路を修正するという方針が，組込みシステム開発に特有のコツであると言える．

初学者は，序盤で回路とディスプレイの注目を繰り返していたことから，誤り箇所を推測できず，修正の方針が定まらない状態に陥っている．初学者の一部にシステムの修正方針についてヒアリングを行ったところ，「色々な要素を見てみないと，どこを修正していいかわからなかった.」という回答や，「どちらから修正を始めるか迷って考えた結果，回路を先に修正した.」という回答が得られた．初学者がデバッグの方針を初めに立てられなかったことから，初学者のデバッグ効率を向上するために，熟練者の修正方針を教示することが有効だと考えられる．

注目物体の時間分布において，熟練度によって違いが表れたことから，それぞれの熟練度においてデバッグ過程を分節化できる可能性がある．例えば，熟練者は回路の修正，ソースコードの修正，システムの正誤確認という三つのフェーズで修正を進め，初学者は回路の修正，ソースコードの修正という二つのフェーズで修正を進めるというように分割できる可能性がある．実験時間を三分割にして注目時間分布や遷移確率を見た場合，遷移確率に有意な差が表れたため，実験時間を正しく分割して分析することで，デバッグのフェーズの意味を明らかにできると考えられる．

序盤において，被験者は問題用紙を中心に作業を進める．熟練度に関わらず，序盤ではシステムの仕様と誤りの確認作業が行われることがわかる．熟練者が中盤で問題用紙から回路への遷移確率よりも，ディスプレイへの遷移確率が高くなったことから，熟練者は回路の修正を序盤で切り上げてソースコードの修正に取り掛かったことがわかる．すなわち，熟練者は中盤においてソースコードを中心に修正を行なっている．終盤のマルコフモデルにおいて，熟練者の回路とディスプレイ間の遷移確率の値が近くなったのは，双方を見比べることでシステムの動作を確認しようとしたためだと推測できる．問題用紙からディスプレイと回路に向かう遷移確率が近いことからも，熟練者がソースコードと回路に誤りがある可能性を等しく疑っていることがわかる．終盤のマルコフモデルにおいて，初学者の回路からディスプレイに向かう遷移確率が高いことから，システムが正常に動かない際にソースコードを疑う傾向があり，回路に誤りがある可能性が低いと判断したためだと考えられる．問題用紙からディスプレイへの遷移確率が高いことからも，初学者がシステムの動作確認でソースコードを重視する傾向があることがわかる．

5. おわりに

本研究では，組込みシステム開発のデバッグにおける視線動作について扱い，熟練度ごとに視線動作の違いがあるかを明らかにするための被験者実験を行った．実験の結果，実験時間全体をまとめて定量分析する方法では熟練度の差が現れないため，熟練者の持つコツを見つけるには，時系列データを傾向ごとに適切に分割して分析する必要があると考えられる．熟練者はデバッグの序盤で回路の修正，中盤でソースコードの修正，終盤でシステムの動作確認を行うという順番の修正方針を立てる．初学者は誤り箇所の特定ができず，修正の方針を定められない傾向があり，実験全体を通して前半で回路の修正，後半でソースコードの修正をすることが判明した．終盤においてシステムが正常に動作しているかを調べる際，熟練者は各要素の誤りがある可能性を等しく疑う傾向があり，初学者はソースコードに誤りがある可能性を疑う傾向があることを確認した．

現段階の分析結果では，デバッグのフェーズに関して，作業内容として意味があるような分割ができていない．したがって，今後の展望として隠れマルコフモデルを用いた

状態推定により，デバッグ作業の内容に基づく分割ができるかを検証する．これにより，熟練者の修正方針を正確に把握することができ，組込みシステム開発のデバッグにおける効率的な手順を明らかにできる．

本研究により，熟練度の違いによるデバッグ方法の違いを明らかにできれば，初学者に対するアドバイスを具体化し，組込みシステム開発特有のコツを伝えられると考えられる．また，初学者のデバッグ効率を向上させるために，具体化されたアドバイスを教示することが，有効であるかを検証できると考えられる．デバッグ効率が向上した場合，組込みシステム開発において，初学者へのコツの教示による学習支援が有効であると判断できる．

参考文献

[1] みずほ情報総研株式会社：IT 人材需給に関する調査，みずほ情報総研株式会社（オンライン），入手先 ⟨https://www.meti.go.jp/policy/it_policy/jinzai/houkokusyo.pdf⟩（参照 2019-10-12）．
[2] 野村総合研究所：IoT 市場の拡大と、日本における IoT 活用のありかた，野村総合研究所（オンライン），入手先 ⟨http://www.nri.com/-/media/Corporate/jp/Files/PDF/knowledge/publication/m_review/2017/nmr38/nmr38-1.pdf?la=ja-JP⟩（参照 2019-10-20）．
[3] 岡本雅子，村上正行，吉川直人：教養教育としての組み込み系プログラミング教材の開発と評価，コンピュータと教育 (CE)，Vol. 2010, No. 6, pp. 1–7 (2011).
[4] 木下大輔，山本椋太，阿部　司，大西孝臣，三上　剛，吉村　斎：組込みシステム入門者向け教材の作成，精密工学会学術講演会講演論文集，Vol. 2015S, pp. 911–912 (2015).
[5] 大野健彦：視線から何がわかるか，認知科学， Vol. 9, No. 4, pp. 565–579 (2002).
[6] 檜山　敦，浅田和宏，並木秀俊，宮廻正明，廣瀬通孝：伝統技能継承のための主観視点を含んだ支援映像の生成，ヒューマンインタフェース学会論文誌，Vol. 12, No. 3, pp. 249–258 (2010).
[7] Olsen, A.: The Tobii I - VT fixation filter (2012).
[8] 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変化，研究報告ソフトウェア工学 (SE)，Vol. 2014-07-02, No. 1, pp. 1–8 (2014).
[9] Tobii: 売上 15%増に結びついたパッケージデザイン調査，Tobii（オンライン），入手先 ⟨https://www.tobiipro.com/ja/fields-of-use/marketing-consumer-research/customer-case/strategir/⟩（参照 2019-11-05）．
[10] Tobii: Eye tracking TC commercial helped Team Foods recover market share, Tobii (online), available from ⟨https://www.tobiipro.com/fields-of-use/marketing-consumer-research/customer-cases/yanhaas/⟩ (accessed 2019-11-05).
[11] 阪井　誠，中道　上，島　和之，中村匡秀，松本健一：WebTracer：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] CRESCENT: 画像解析ソフトウェア Faceware Analyzer，CRESCENT（オンライン），入手先 ⟨https://www.crescentinc.co.jp/product/faceware/p_syosai/⟩（参照 2019-11-06）．
[13] 江川　陽，白山　晋：視線分析手法の高度化とその応用，人工知能学会全国大会論文集，pp. 1–4 (2009).
[14] 志賀優毅，内海ゆづ子，岩村雅一，カイクンツェ，黄瀬浩一：一人称視点画像を用いた文書画像の分類，研究報告コンピュータビジョンとイメージメディア（CVIM），Vol. 2014-CVIM-192, No. 15, pp. 1–7 (2014).
[15] Orlov, P.: Primary investigation of applying Hidden Markov Models for eye movements in source code reading., pp. 18–20 (2015).
[16] Busjahn, T., Schulte, C., Sharif, B., Simon, B., Begel, A., Hansen, M., Bednarik, R., Orlov, P., Ihantola, P., Alperovich, G. and Jetbrains, M.: Eye Tracking in Computing Education, In Proceedings of the tenth annual conference on International computing education research, pp. 3–10 (2014).
[17] 花房　亮，山岸秀一，松本慎平，加島智子：機械学習処理に基づいたプログラミング読解中の視線軌道の自動分類，人工知能学会全国大会論文集，Vol. 2015, pp. 1–2 (2015).
[18] 松本光稀，若原　徹：視線情報の分析に基づくプログラミング教育支援システムの提案，第 80 回全国大会講演論文集，Vol. 2018, No. 1, pp. 721–722 (2018).
[19] 西方真弓，西原亜矢子，定方美恵子，清野由美子，井越寿美子，笠井美香子，佐藤富貴子，川合　功，坂本　信，小浦方格，田邊裕治：新人看護師の ‘観察・判断への気づき’ を育てる視線解析を用いた教育プログラムの評価: 臨床経験豊富な看護師の ‘DVD 教材’ 視聴による気づきの分析，新潟大学保健学雑誌，Vol. 11, No. 1, pp. 25–32 (2014).
[20] 早川和輝，長谷川大，佐久田博司：主観視点の 3D 手本動作教材提示によるドラム演奏学習支援および熟練者視線情報を利用した教材による学習効果，知能と情報，Vol. 28, No. 1, pp. 511–521 (2016).
[21] Redmon, J. and Farhadi, A.: YOLOv3: An Incremental Improvement (2018).