卒　業　研　究　報　告　書

平成29年度

研究題目

生体データと作業履歴に着目した
プログラム理解過程の分析

指導教員　　上　野　秀　剛　　准教授

氏　　名　　石　田　豊　実

平成30年02月16日　提出

奈 良 工 業 高 等 専 門 学 校　　情 報 工 学 科

生体データと作業履歴に着目した
プログラム理解過程の分析

上野研究室　石田豊実

プログラム理解やデバッグにおいて，プログラマの状態を計測する方法として脳波や視線が用いられている．脳波は人の精神状態との関連があるとされており，視線はプログラム理解やデバッグを行う際の傾向を見るのに用いられている．脳波と視線はプログラマの状態を定量的に分析するのに有効な手段であるとされている．プログラマの状態を把握するために脳波や視線の計測を行う研究は多く行われている．しかし，バグを探しているプログラマの状態を把握するために脳波と視線を同時計測した研究は行われていない．よって本研究では，バグを探しているプログラマの状態を知るために脳波と視線を計測し分析する．バグを発見するためには，ソースコード中の間違いに気づかなければならないが，そのためには仕様やソースコードを理解している必要がある．本研究ではバグを探しているプログラマの状態をプログラムの動作を理解しているかとバグを正しく判断出来るかの2つに分ける．2つの状態に対し，脳波と視線を分析し違いがあるか検証する．実験では，脳波の周波数成分であるα波とβ波を計測指標とし，プログラム動作を理解している時とそうでない時で差があるか調べる．また，バグを正しく判断できる時とそうでない時に差があるかも調べる．実験では被験者にタスクを与え，プログラム動作を理解するステップとバグを判断するステップの脳波と視線を計測する．プログラム動作理解ステップで計測した脳波を分析し，プログラム動作を理解できた時とそうでない時のαやβの大きさを比較し考察する．バグ判断ステップで計測した脳波を分析し，バグを正しく判断できた時とそうでない時のαやβの大きさを比較し考察する．実験の結果，プログラム動作理解ステップでは動作を理解した時は動作を理解していない時に比べ，αが有意に大きくなった．バグ判断ステップでは正しく判断できた時は正しく判断できなかった時に比べ，βが有意に大きくなった．この結果は，バグを探しているプログラマの状態を把握するための手がかりになると考えられる．

目 次

1 はじめに **2**

2 関連研究 **3**

2.1 脳波と心理状態 3

2.2 脳活動計測を用いたプログラム理解の研究 3

2.3 視線と熟練度 4

2.4 視線計測を用いたデバッグ・プログラム理解の研究 4

3 脳波と視線 **5**

3.1 脳波 5

3.1.1 計測方法 5

3.1.2 周波数が示す特徴 5

3.2 視線 6

3.3 脳波と視線の同時計測 7

4 実験 **8**

4.1 実験環境 8

4.1.1 脳波計測環境 8

4.1.2 視線計測環境 9

4.2 タスク 10

4.2.1 プログラム動作理解 10

4.2.2 バグ判断 12

4.3 実験の手順 14

4.4 分析の手順 15

5 結果と考察 **17**

5.1 プログラム動作理解ステップ中の脳波 17

5.2 バグ判断ステップ中の脳波 18

5.3 時間による正規化 19

6 おわりに **22**

謝辞 **24**

参考文献 **25**

1 はじめに

ソフトウェアの開発過程で混入するバグを発見･除去する作業は開発工程に遅れを生じる．効率良くバグを発見できれば除去する作業にも早く取り掛かることができるため，開発工程の遅れを少なくできる．しかし,バグを効率良く発見できないプログラマは存在する．そういったプログラマには効率良くバグを発見できるよう支援を行う必要がある．

ここで，効率よくバグを発見できるような支援をより具体的に考えるためにプログラマがバグを発見するまでの過程を考えてみる．データフローに関わるバグはプログラムを理解してなければ発見できない．また，仕様とは異なる動きをするバグはプログラムや仕様を理解していなければ発見できない．このようにバグを発見するためにはプログラムや仕様を理解した後バグがあるかを判断する必要があると考える．従ってバグを発見できないプログラマは仕様やプログラムを理解していない等の状態にあると考える. その状態がプログラムの理解不足だった場合はプログラムを理解できるよう支援するべきである．仕様の理解不足だった場合は仕様を理解できるよう支援するべきである．このように支援を行うには，バグを探しているプログラマが仕様やプログラムを理解しているかを判別する必要がある．しかし，プログラムや仕様の理解の有無は本人に直接確認しなければならず, 確認の齟齬が生じる可能性もある．つまり，プログラムや仕様の理解の有無を判別するのは難しいため，バグを探しているプログラマが仕様やプログラムを理解しているどうかを判別する方法が必要である．

脳波や視線はプログラマの状態が反映されるためプログラマの状態を判別する手段として多くの研究で用いられている[1][2][3][15]. 従って本研究ではバグを探しているプログラマの状態を判別するために脳波や視線を計測し分析する．分析して得た知見がバグを探しているプログラマの状態を把握する手がかりになると考える．

2 関連研究

2.1 脳波と心理状態

計測した脳波から人の心理状態を観測する研究がさまざまな研究分野で存在する．武者らは計測された脳波から感情を推定する方法を提案している[5]. 計測に使用した10個の電極の内2個の電極で記録される電位の相互相関関数を特徴量とし，全ての電極の組み合わせから人間の「怒り/ストレス」「喜び」「悲しみ」「リラックス」に関する特徴量を捉えようとした．満倉は，小型の脳波計測器のみで人の感性を取得できる装置を構築し，脳波からストレスを検知するシステムを提案している[7]. 人間の感性（ストレス，興味度，集中度，好き，嫌いなど）を対象に，被験者実験によりパターン認識手法で各感性の強さと脳波の関係を推定した．その結果，11Hzと16Hzの周波数成分が同時に増加することが人間の“嫌”な状態を示すことを明らかにした．

このように脳波には精神状態との関連があると考えられる．プログラムを読んでいる際のプログラマの精神状態は，プログラムの動作を理解している時には落ち着いた状態にあり，逆に動作を理解していない時にはストレスのかかった状態にあると考えられる．バグを判断する際にも，同様に精神状態に違いが現れると考えられる．この精神状態の違いが脳波に現れると考える．

2.2 脳活動計測を用いたプログラム理解の研究

プログラミング作業者の脳活動を計測し，プログラム理解度やプログラマの状態との関連を調査している研究が複数報告されている[1][8]. 脳活動からプログラムの理解度を計測する研究の主な目的は，プログラムの理解が不十分な作業者への支援や作業の効率向上である．Siegmundらは，fMRIを用いてプログラム理解における脳の部位ごとの活性化を調査している[1]．最大18行の短いソースコードを理解するタスクを対象とした実験の結果，問題解決，記憶，および文章理解に関係する脳領域がプログラム理解時に活発になることを示している．中川らは，プログラム理解活動を定量的に評価することを目的に，前頭前野の脳血流を計測することでプログラム理解に困難が生じている状態の判別が可能か検証する実験を行った[8]. 実験により，課題の難易度によって脳活動に差があり，課題の序盤から中盤にかけて脳血流値の正の変化量が最大になると示している．

上記の研究から，プログラム理解度やプログラマの状態は脳活動と関連があるといえる．本研究ではプログラムの動作を理解する際とバグであると判断する際のプログラマの脳波を計測する．動作を理解できたこととバグであると正しく判断できたことが脳波に反映されるか調査する．

2.3 視線と熟練度

視線の計測は初心者と熟練者の違いを分析することを目的に，特に認知科学の分野においてよく用いられている[3][9]. Lawらは腹腔鏡手術の訓練装置を使用している際の初心者と熟練者の視線の動きを分析している[3]. 分析の結果，熟練者は初心者に比べて患部をより集中して見ていることが明らかになっている．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[9]. 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した．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．

本研究が対象とする，バグかどうかの判断において，プログラマは仕様とプログラムの動作が一致しているか判断しなければならない．そのために，仕様やプログラムの理解が必要となる．よってバグを正しく判断している時はそうでない時に比べ，仕様を理解するための視線移動，プログラムを理解するための視線移動等に違いが現れると考える．このようにプログラミングにおいても，プログラム理解やデバッグを効率よく行う人とそうでない人で視線移動に違いが現れるのではないかと考える．

2.4 視線計測を用いたデバッグ・プログラム理解の研究

デバッグやプログラム理解における視線を計測した研究も複数報告されている[4][6]. SteinとBrennanはレビュー作業者に他の作業者がレビューした際の視線を見せることでバグ発見時間が短くなることを明らかにしている[4]. 上野らはバグの探索時，ある変数が使用された行を見ると，その変数を宣言している行やその変数が直近に使用された行に戻って確認することを明らかにし，この視線の動きをさらに分析することでプログラムの理解の過程を明らかにできることを示唆している[6]. プログラムの動作を理解する過程において，理解できている時とできていない時で変数の確認作業や仕様とプログラムを読む割合等の視移動に違いが現れると考えられる．バグかどうか判断する過程においては，正しく判断できている時はバグの原因となっている変数を確認する適切な視線の動きが多くあり，正しく判断できていない時は適切な視線の動きが少ないと考える．

上記の研究からデバッグやプログラム理解の過程を視線計測によって明らかにできると考えられる．本研究ではプログラムの動作を理解する過程とバグかどうかを判断する過程において視線計測を行う．プログラムの動作を理解した時とそうでない時，バグかどうかを正しく判断できた時とそうでない時で視線移動に違いがあるか調査する．

3 脳波と視線

3.1 脳波

山本らは，プログラムの実装戦略が推定できた時の脳波と推定できなかった時の脳波を比較している[15]. 計測指標として，脳波の周波数成分である α 波と β 波を用いている．本研究においても計測指標として脳波の周波数成分を用いるため，脳波の計測方法と周波数成分の特徴を山本らの論文の3章から引用する．

3.1.1 計測方法

脳波とは，脳から生じる電気活動を電位を縦軸，時間を横軸にとって記録したものである[11]. 脳波は，頭皮上に装着した電極から計測される．電極の配置は図3.1に示す国際式10-20電極法に則って行う[10]. 国際式10-20電極法では耳のアースを除き19箇所の装着位置が指定されており，検査や研究の目的によって使用する電極を決定する．

脳波の導出法には主に基準電極導出法と双極導出法の2種類の方法がある．基準電極導出法では，脳電位の電場内に装着した計測用電極と，電場外に装着した基準電極の2つの電極の電位差として脳電位を測定する．双極導出法は，基準電極を用いず，2つの計測用電極を脳電位の電場内に置いて記録する方法である．脳電位は2つの電極の電位差として測定される．計測用電極の電極間隔が狭い場合に電位差を計測する際は基準電極導出法を，優勢な背景成分を除去して部位差を強調する目的で計測する際には双極導出法を選択する[12].

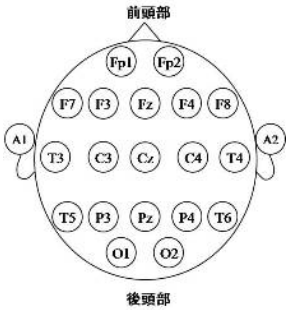

図 3.1: 国際式 10-20 電極法

3.1.2 周波数が示す特徴

一般的に脳波は時間の経過にともなって複雑な電位変動を示す不規則な振動現象とみなされ，高速フーリエ変換(FFT)を用いてパワースペクトルが求められる．脳波からパワースペクトルを求める際には，国際脳波学会によって周波数帯

域ごとに付けられた分類および名称を用いる[13]. 各帯域の名称と周波数帯域を以下に示す.

- δ 波： 0.5〜4Hz
- θ 波： 4〜8Hz
- α 波： 8〜14Hz
- β 波： 14〜30Hz
- γ 波： 30Hz 以上

δ 波や θ 波は睡眠状態にあるときに出現する. α 波は安静状態にあるときに強く表れる周波数帯域で，リラックスし，何かに没頭しているときに出現する. 他の周波数帯域の波と比べて振幅も連続性も最も高い. 眠気を感じるなど覚醒が低下してくると，α 波の振幅が低下して不連続になる. また，α 波は開眼すると大幅に減少し，閉眼すると再び出現する. 一般的にこれを α 波減衰と呼ぶ. また，緊張や不快な感情を抱いているときや日常の思考状態では β 波が出現する. γ 波は，不安で興奮しているときに出現する[12][13]. これらの周波数帯域の内，α 波と β 波はリラックス状態や精神活動状態によって変動するとされており[13]，さまざまな作業における人間の心理状態の計測指標に用いられている[14].

3.2 視線

視線の計測方法には，計測装置を眼球に接触させるもの（接触型）と接触させないもの（非接触型）がある. 本研究では，被験者への身体的な負担が少ない非接触型の計測装置を用いる. 非接触型の視線計測装置は，被験者の目に弱い赤外線を当てた時にできる反射点と瞳孔の位置を記録し眼球の向きを計測する. 被験者によって違う眼球の大きさ等の特徴や，周囲の照明環境によって赤外線の反射や瞳孔の見え方が変わるので，視線計測する前には，被験者ごとにキャリブレーション（補正）を行う.

視線計測を行うことによって，プログラマの状態によって視線移動に違いがないか分析することができる. 例えば，プログラム動作を理解できる時とそうでない時に視線移動に違いがあればプログラム動作理解におけるプログラマの状態を明らかにする手がかりになると考えられる. またバグを正しく判断できる時とそうでない時に視線移動に違いあればバグ判断におけるプログラマの状態を明らかにする手がかりになると考えられる. よって本研究では，プログラム動作理解とバグ判断の過程で視線を計測する.

3.3 脳波と視線の同時計測

2章で述べたこれまでの研究からプログラム理解やデバッグにおいて脳波・視線の計測が効果を持っていることが分かる．本研究において，バグを探しているプログラマの状態をまず脳波を分析することで明らかにする．しかし，脳波を分析し得た結果が仮説と違った時，その根拠を視線から説明できると考える．例えば，バグを正しく判断できた時には落ち着いた状態にあるのでα波優位になるという仮説を立てる．もし，分析結果としてβ波優位であった場合，その理由として新たに仮説を立てる．バグを正しく判断できた時に特徴的な視線移動を発見できればそれを根拠に脳波の分析結果を説明できると考える．また，脳波の分析結果が仮説通りであった時でも視線と組み合わせることでさらに詳細な分析ができると考える．よって，脳波の分析結果の根拠を探る手段としても視線計測は有効ではないかと考える．

本研究において脳波計測は，バグを探しているプログラマをプログラム動作理解の有無やバグ判断の正誤に分類した時に違いが現れるか調べるために用いる．視線計測は，計測した脳波の分析結果の根拠を探る手段として用いる．プログラム動作理解の有無で脳波に違いがあった時，その時にどんな視線移動があるのか調べるために脳波と視線を同時計測する．

プログラム理解における脳波や視線を同時計測した研究も報告されている．Fritzらはプログラム理解時の視線，脳波，筋電を同時計測し，機械学習を用いて計測値からタスクの性質やプログラマの状態の推定を試みている[2]．Fritzらの研究から脳波や視線を個別に計測するだけでなく，同時計測する事も有効である事が分かる．Fritzらは，タスクの性質やプログラマがタスクに難しさを感じているかの推定精度を脳波，視線，筋電の同時計測によって向上させることを目的としている．本研究では，バグを探しているプログラマの状態を脳活動や視線移動の違いを発見することで明らかにすることを目的としており，違いがある．

4 実験

プログラム課題を被験者に提示し，その内容を理解している間の脳波と視線を計測する．被験者は奈良工業高等専門学校の学生16人で，年齢は19歳から20歳，全員がJavaによるプログラミングの基礎講義を受講済みである．

4.1 実験環境

実験は被験者1名と実験者2名のみが居る静かな部屋で実施する．体動によるアーチファクトを抑えるために，ひじ掛け･足置きを備えた椅子に座り，頭部と体をできるだけ動かさないように指示する．実験に使用するのは，脳波計測装置，視線計測装置，タスク提示用PC，脳波計測用PC，記録用PCである．脳波計測装置にはナノテックイメージ社製Nexus-10 MARK IIを用いる．視線計測にはTobii社製tobii Eye Tracker 4Cを用いる．

4.1.1 脳波計測環境

脳波計測に用いるハードウェアは，ナノテックイメージ社製Nexus-10 MARK IIと計測用のノートPC1台である．Nexus-10 MARK IIは計測周期は256Hzであり，計測された脳波はBluetooth軽油で脳波計測用PCに転送され，CSVファイル形式で出力される．計測用のノートPCは，Windows 7 Intel(R) Core(TM)i5-3380M 2.90GHzでメモリ搭載量は4GBである．図4.1に装置の外観と装着時の様子を示す.

(a) 装置の外観

(b) 装着時の様子

図 4.1: 脳波計測用装置

脳波計測に用いるソフトウェアは，Nexus-10 MARK IIに対応したソフトウェアであるBioTrace+ Softwareである．ソフトウェアはBluetoothによってNexus-10 MARK IIから脳波データをリアルタイムで受信し，記録する．CSVファイル形式で記録した脳波を出力できる．図4.2は計測時の動作画面である．上半分が脳波の生データであり，リアルタイムで脳波が表示される．左下のSTART，および右下のENDボタンは課題の開始･終了時を記録するためのボタンで，出力されるCSVファイルに「Start」または「End」が記録される．

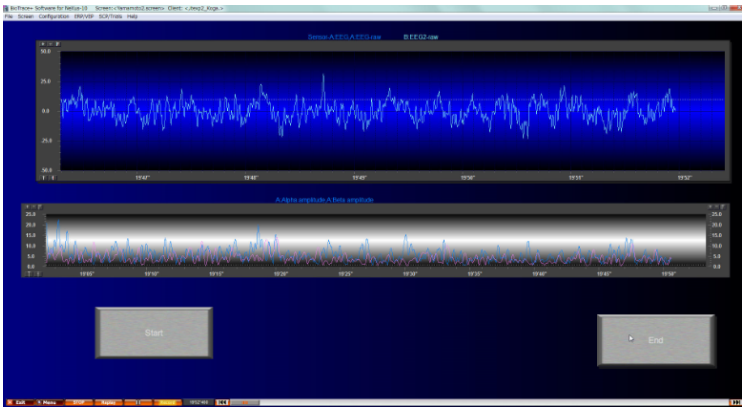

図 4.2: BioTrace+動作画面

4.1.2 視線計測環境

視線計測に用いるハードウェアは，Tobii 社製の Eye Tracker 4C とタスク提示用のノート PC である．タスク提示用のノート PC は，Windows 10 Intel(R) Core(TM) i5-6200U CPU @ 2.30Hz でメモリ搭載量は 8GB である．である．Eye Tracker 4C は非接触型の視線計測装置であるため，被験者にかかる負担を軽減することが出来る．タスク提示用 PC のディスプレイの解像度は横 1920*縦 1080 画素である．

図 4.3 に装置の外観を示す.

図 4.3: 装置の外観

視線計測に用いるソフトウェアは 2 つあり，1 つ目は Tobii 社が配布している設定ツールである．設定ツールから視線のキャリブレーションを行う.

2 つ目のソフトウェアは上野准教授が作成した実験ツールである．実験ツールはタスクの提示を行う．タスクは 3 つのタブに分けて提示する．タスク内容は 4.2 で説明する．また，実験ツールは tobii Eye Tracker 4C から得た視線データを記録し，CSV ファイルで出力する．キーボードの space キーでタスクの表示･非表示の切り替え，右キー，左キーでタブの切り替えが出来る．実験時には，できるだけ体動を抑えるために space キーと右，左キーをそれぞれボタンに割り当てたコントローラを使用する．タスク提示ツールの動作画面を図 4.4 に示す.

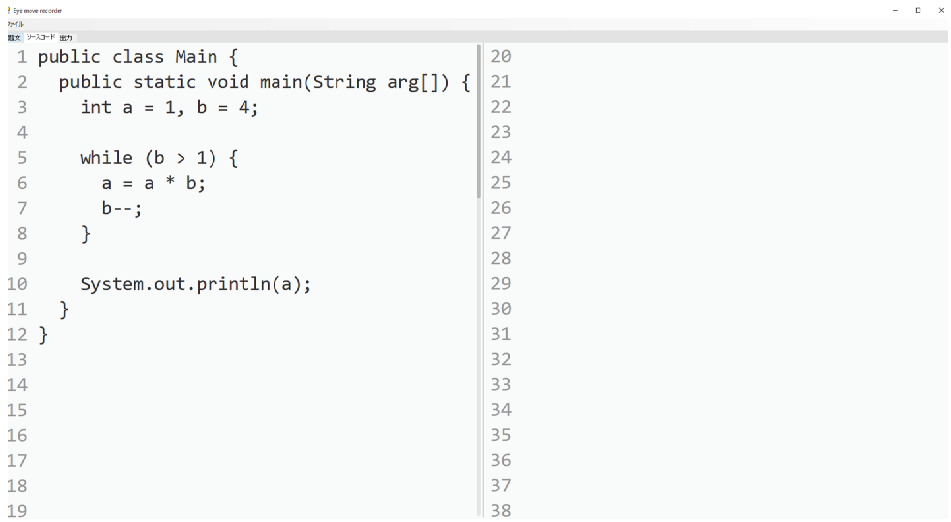

図 4.4: タスク提示ツール動作画面

4.2 タスク

実験で課すタスク内容について説明する．タスクは被験者1人につき16問与える．1タスクは2つのステップに分かれている．1つ目のステップは，プログラム動作を理解するステップである．2つ目のステップは，バグかどうかを判断するステップである．以降で2つのステップの内容を説明する．

4.2.1 プログラム動作理解

ここではタスク中の1つ目のステップであるプログラム動作理解についての説明をする．このステップでは，被験者にタスクを見せ，ソースコードの動作を理解しているか確認するための問いに口頭で回答してもらう．回答内容が事前に用意した解答と一致していればソースコードの動作を理解しているとみなす．一致していなければソースコードの動作を理解していないとみなす．また，このステップには制限時間があり，一定時間を超えるとソースコードの動作を理解していないとみなす．回答内容が解答と一致しているかは被験者には伝えない．被験者は制限時間以内に出力フォームの問いが分かったら，spaceキーを割り当てたコントローラのボタンを押して回答する．回答する，または制限時間が過ぎたらバグ判断ステップを行う．バグ判断ステップについては4.2.2で説明する．

被験者に提示するものは，プログラムの仕様，ソースコード，出力フォームの3つで，それぞれタスク提示ツール上の異なるタブに表示される．図4.5にプログラム動作理解ステップに使用したプログラムの仕様，ソースコード，出力フォームの例を示す．

(a) プログラムの仕様

```
public class Main {
  public static void main(String arg[]) {
    int a = 1, b = 4;

    while (b > 1) {
      a = a * b;
      b--;
    }

    System.out.println(a);
  }
}
```

(b) ソースコード

(c) 出力フォーム

図 4.5: プログラム動作理解ステップの提示物

図4.5の説明をする．プログラムの仕様には，図4.5(a)のようにプログラムの概要が日本語で書かれている．ソースコードには，図4.5(b)のようにプログラムの仕様に合ったソースコードが書かれている．ソースコードは全てJava言語を用いて作成する．出力フォームには，図4.5(c)のようにソースコードの動作が理解できたか確認するための問いが日本語で書かれている．図4.5(c)の場合の正答は12である．

本研究では，プログラム動作を理解している時と理解していない時の脳波と視線を分析する．そのためプログラム動作を理解したグループと理解していないグループに分け，統計的な分析を行う．その際，どちらかのグループにデータ数が偏ってしまうと片方のグループのデータ数が不足し，統計手法を用いることができなくなる．よって，データ数が偏らないようにタスクを設定したい．

プロフラム動作を理解するのが簡単なタスクは理解している状態になりやすいと期待できる．反対にプログラム動作を理解するのが難しいタスクはプログラム動作を理解していない状態になりやすいと期待できる．よって，低難易度のタスクと高難易度のタスクを半分ずつ用意すれば，理解している状態と理解していない状態を均等に分けることができるはずである．よって，本実験では低難易度と高難易度のタスク数は8問ずつとする．低難易度のソースコードは全てmainメソッドのみを使用する．1重の繰り返し文や条件分岐で構成された理解が容易と思われるソースコードを使用する．高難易度のソースコードは複数メソッドの使用や再帰構造によって制限時間以内での理解が難しいと期待できる複雑なアルゴリズムを使用する．

タスクとして提示する課題の一覧を表1に示す．各被験者に提示する問題の順番は，順序効果を考慮しカウンターバランスを行う．カウンターバランスとは提示するタスクの順番が被験者全体でつり合いが取れるよう被験者毎にタスクの順番を変えることである．

タスクには予備実験を基に決定した制限時間を設定する．予備実験は低難易度のソースコードを理解するのには十分で，かつ，高難易度のソースコードを理解するには不十分な時間を測定するために行った．予備実験では2人の被験者を対象に，低難易度のタスクから最も難しいと思われるタスクを2つ，高難易度のタスクから最も簡単と思われるタスクを2つ行ってもらい所要時間を計測した．この予備実験に使用したタスクは本実験には使用しない．予備実験の結果を基に，制限時間を2分30秒とした．

4.2.2 バグ判断

ここではタスク中の2つ目のステップであるバグ判断についての説明をする．

このステップでは，被験者にプログラム動作理解ステップで見せたソースコードの一部を改変したソースコードを見せる．改変後のソースコードには，仕様を

表 1: 実験に使用したタスクの一覧

	ファイル名	仕様
1	Factorial.java	階乗の計算
2	SearchMax.java	最大値検索
3	PrimeNum.java	素数判定
4	SearchMedian.java	中央値検索
5	Power.java	累乗計算
6	Swap.java	2つの数値の入れ替え
7	Contained_Substring.java	指定した文字列が含まれているか判定
8	ReverseString.java	文字列を反転させる
9	TowerOfHanoi.java	ハノイの塔
10	NumOfRoute.java	経路数を求める
11	MakePermutation.java	順列を全列挙する
12	Combination.java	組み合わせを漸化式から求める
13	PayMoney.java	支払う硬貨の組み合わせを求める
14	StrCombination.java	文字列の組み合わせを求める
15	JOI14pre3.java	2014年情報オリンピック予選　問題3
16	lcm_gcd.java	最小公倍数と最大公約数を求める

満たすものと仕様を満たさないもの（バグ）があり，被験者には改変後のソースコードが仕様を満たすか，満たさないか判断し解答してもらう．被験者は仕様を満たすと判断した場合は”満たす”，仕様を満たさないと判断した場合は”満たさない”と口頭で回答してもらう．回答が事前に用意した解答と一致していれば，正しく判断できたとみなす．一致していなければ，正しく判断できなかったとみなす．また，このステップには制限時間があり，一定時間を超えると正しく判断できなかったとみなす．回答が解答と一致しているかは被験者には伝えない．被験者は制限時間以内に仕様を満たすかどうか分かったら，spaceキーを割り当てたコントローラのボタンを押して回答する．回答をする，または制限時間が過ぎたら別のタスクのプログラム動作理解ステップを行う．

被験者に提示するものは，プログラムの仕様，改変後のソースコード，出力フォームの3つである．プログラムの仕様と出力フォームはプログラム動作理解ステップと同様のものを提示する．改変後のソースコードとは，プログラム動作理解ステップで使用したソースコードの一部を改変したソースコードである．

本研究では，バグであると正しく判断できた時と正しく判断できなかった時の脳波と視線を分析する．そのためバグであると正しく判断できたグループとできなかったグループに分けて分析を行う．バグか否かを判断する難しさはソースソースコードの難易度に影響を受けると考えられる．そのため，低難易度，高難易度それぞれの半分について，変更後のソースソースコードが仕様を満たさないよう設定する．

本ステップにおいても，プログラム理解のステップと同様に予備実験の結果を基に1分の制限時間を設定する．被験者は改変後ソースコードが仕様を満たすかどうか分かったら，spaceキーを割り当てたコントローラのボタンを押して回答する．制限時間を超えると正しく判断できなかったものとする．本研究ではバグを正しく判断できるかを計測したいため，改変部分を探す作業に時間がかからないよう，改変部分には色付けを行う．図4.6に改変前後のプログラムの例を示す．

```
1 public class Main {
2   public static void main(String arg[]) {
3     int a = 1, b = 4;
4
5     while (b > 1) {
6       a = a * b;
7       b--;
8     }
9
10    System.out.println(a);
11  }
12 }
13
14
15
16
17
18
19
```

(a) 改変前ソースコード

```
1 public class Main {
2   public static void main(String arg[]) {
3     int a = 1, b = 4;
4
5     while (b >= 1) {
6       a = a * b;
7       b--;
8     }
9
10    System.out.println(a);
11  }
12 }
13
14
15
16
17
18
19
```

(b) 改変後ソースコード

図 4.6: 改変前後のソースコード

4.3 実験の手順

実験の順序を以下に示す．

1. 実験の説明
   実験概要，脳波・視線計測時の注意点を説明する．

2. 装置の装着・設定
   脳波計測装置NeXus-10 MARKIIを被験者に，視線計測装置Eye Tracker 4CをPC

に装着し，設定する．

3. 練習問題
   1タスクの流れを被験者に確認してもらうために，練習タスクを実施してもらう．

4. 動作理解タスクの実施
   ディスプレイにタスクを2分30秒間提示し，プログラムの動作を理解してもらう．

5. 解答
   被験者には動作を理解した時にコントローラのボタンを押して答えを口述してもらう．

6. 改変後タスクの実施
   ディスプレイに改変後のタスクを1分間提示し，仕様を満たすか判断してもらう．

7. 解答
   被験者には仕様を満たすか判断できた時にコントローラのボタンを押して答えを口述してもらう．

8. 全タスクの実施
   1タスク当たり手順4〜 7 を1回を行う．全タスクである16タスクを行うため，手順4〜7を16回繰り返す．

4.4 分析の手順

プログラム動作理解ステップとバグ判断ステップそれぞれについて，タスク中の脳波を分析する．脳波データは被験者1人当たり16タスク*被験者数のデータがプログラム動作理解ステップ，バグ判断ステップの両方で計測される．得られた脳波データに対して個々にFFTをかけていき，α 波の帯域である8Hz〜13Hz，β 波の帯域である14Hz〜30Hzのそれぞれの帯域の成分抽出を行う．脳波は個人差が大きいため，抽出された成分データを各被験者の平均値で正規化する．

正規化後のパワースペクトルを，プログラム動作理解ステップで問いに正解しているものとそうでないもので2つのグループに分ける．2つのグループ間で正規化後の α 波（以降，単に α とする．）を計測指標とし，α の大きさを比較し分析する．同様に正規化後の β 波（以降，単に β とする．）も計測指標として β の大きさを比較し分析する．

またバグ判断ステップで正しくバグかどうか判断できているものとそうでないもので2つのグループに分ける．2つのグループ間で α と β を計測指標としてそれぞれで大きさを比較し分析する．分析には統計処理を用いる．

プログラムの動作を理解している時には精神的負荷が少ない状態にあると考えられるため，α は大きく，β は小さくなると考えられる．反対にプログラムの動作を理解していない時には精神的負荷が多い状態にあると考えられるため，α は小さく，β は大きくなると考えられる．同様の理由で，仕様を満たすと正しく判断できた時には α は大きく，β は小さくなると考えられる．仕様を満たすと正しく判断できなかった時には α は小さく，α は大きくなると考えられる．

5 結果と考察

16人の被験者を対象に実験を行ったが，システムトラブルがあったため11人の脳波データを分析することが出来なかった．よって5人の脳波データを分析した．

5.1 プログラム動作理解ステップ中の脳波

動作を理解できた時と，できなかった時の2グループについて，タスク中のα，βの値を図5.1に示す．箱の中央付近の横線はデータの中央値を表し，箱の横線は

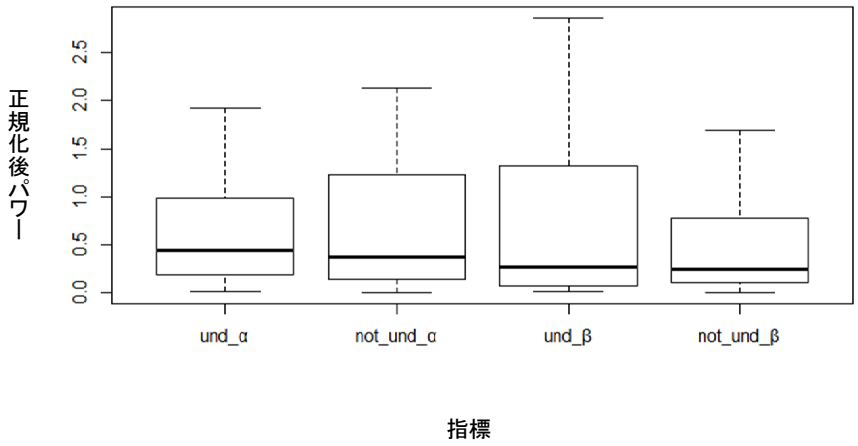

図5.1: プログラム動作理解ステップ中の脳波の比較

下側がデータの第1四分位数，上側が第3四分位数を表す．箱の上下の短い横線は下側がデータの最小値，上側がデータの最大値を表す．以降，プログラム動作を理解した状態をund，見当がつかなかった状態をnot_undとする．und，not_undの各状態についてαを比べる．undでは，中央値が0.436，平均値が0.939であった．not_undでは中央値が0.371，平均値が1.09であった．つまり，プログラム動作理解ステップ中のαを比べると，中央値はundの方が，平均値はnot_undの方がαが大きかった．中央値と平均値で異なる傾向が見られた．

βも同様に比べる．undでは，中央値が0.462，平均値が1.15であった．not_undでは中央値が0.259，平均値が0.239であった．つまり，プログラム動作理解ステップ中のβを比べると，中央値，平均値共にundの方がβが大きかった．

以上の結果について統計的な有意差があるか確かめるに検定を行った．データの等分散性はないものとして，Welchのt検定を行った．und，not_undでのαについてp=0.661，βについてp=0.304とどちらも有意差は見られなかった．

5.2 バグ判断ステップ中の脳波

バグを正しく判断できた時と，できなかった時の2グループについて，タスク中のα，βの値を図5.2に示す．以降，バグを正しく判断できた状態をdiff，正しく

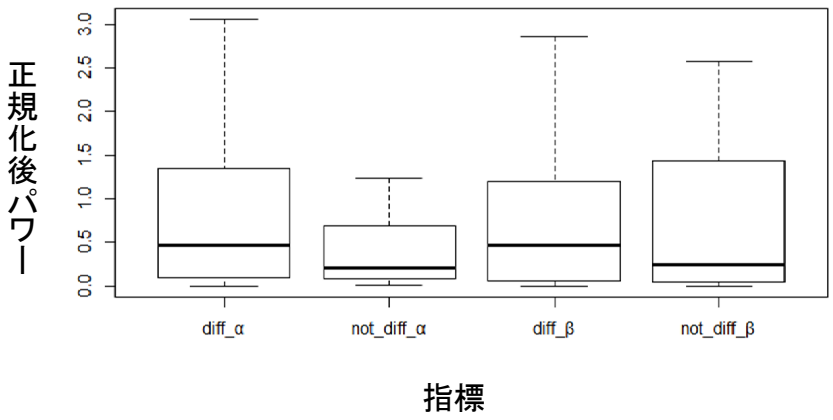

図5.2: バグ判断ステップ中の脳波の比較

判断できなかった状態をnot_diffとする．diff，not_diffの各状態についてαを比べる．diffでは，中央値が0.468，平均値が1.01であった．not_diffでは中央値が0.203，平均値が0.949であった．つまり，バグ判断ステップ中のαを比べると，中央値，平均値共にdiffの方がαが大きかった．

βも同様に比べる．diffでは，中央値が0.462，平均値が1.08であった．not_diffでは中央値が0.250，平均値が0.737であった．つまり，バグ判断ステップ中のβを比べると，中央値，平均値共にdiffの方がβが大きくなった．

等分散性はないと仮定したWelchのt検定による検定の結果，αでp=0.893，βでp=0.329とどちらも有意差は見られなかった．

5.3 時間による正規化

5.1, 5.2で有意差が得られなかった原因はタスクにかかった時間にあると考える．FFTでは信号の中から求めたい周波数帯域のパワースペクトルを時間で積分していく．よって時間が長ければ長い程パワースペクトルは高くなると考える．タスクを終えるのに要した時間（以降，タスク所要時間とする）は，プログラム理解ステップでは最小27.24秒，最大150秒であった．バグ判断ステップでは最小6.86秒，最大60秒であった．このタスク所要時間の差が算出したパワースペクトルに影響を与えていると考えられる．そこでそれぞれのタスク所要時間で正規化してパワースペクトルを求める．

まず，プログラム動作理解ステップでの結果を示す．プログラム動作理解ステップでは，16*5（1人当たりのタスク数*被験者数）タスク中2タスクで他のタスクに比べαが100倍以上大きなものがあった．このタスクが結果に影響を与えている可能性があったため，これを除去した．以降，時間でタスク所要時間で正規化したαとβのパワースペクトルを，tnorm_α，tnorm_βとする．undと，not_undについて，タスク中のα，βの値を図5.3に示す．und，not_undの各状態についてtnorm_

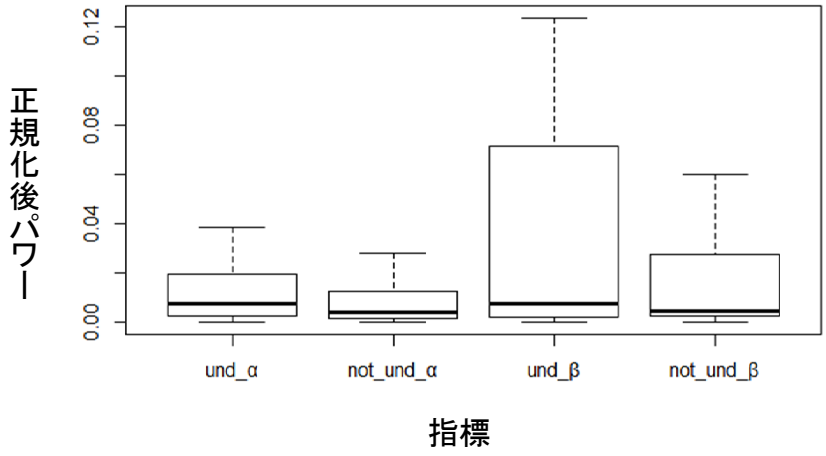

図5.3: プログラム動作理解ステップ中の脳波の比較(時間正規化後)

αを比べる．undでは，中央値が0.007，平均値が0.016であった．not_undでは中央値が0.004，平均値が0.007であった．つまり，プログラム動作理解ステップ中のtnorm_αを比べると，中央値，平均値共にundの方がtnorm_αが大きかった．

tnorm_βも同様に比べる．undでは，中央値が0.007，平均値が0.083であった．not_und

では中央値が0.004，平均値が0.044であった．つまり，プログラム動作理解ステップ中のtnorm_βを比べると，中央値，平均値共にundの方がtnorm_βが大きかった．

以上の結果が偶然に起こったものでないか確かめるために検定を行った．データの等分散性はないものとして，Welchのt検定を行った．und，not_und間のtnorm_αについてt検定を行うとp値は0.024，tnorm_βについてt検定を行うとp値は0.214となり，tnorm_αについて有意差が見られた．

以上の結果からプログラマがプログラムの動作を理解している時は，プログラムの動作を理解していない時よりもα波のパワースペクトルが大きくなることが示された．

次に，バグ判断ステップでの結果を示す．diffと，not_diffについて，タスク中のtnorm_α，tnorm_βの値を図5.4に示す．diff，not_diffの各状態についてtnorm_αを比べ

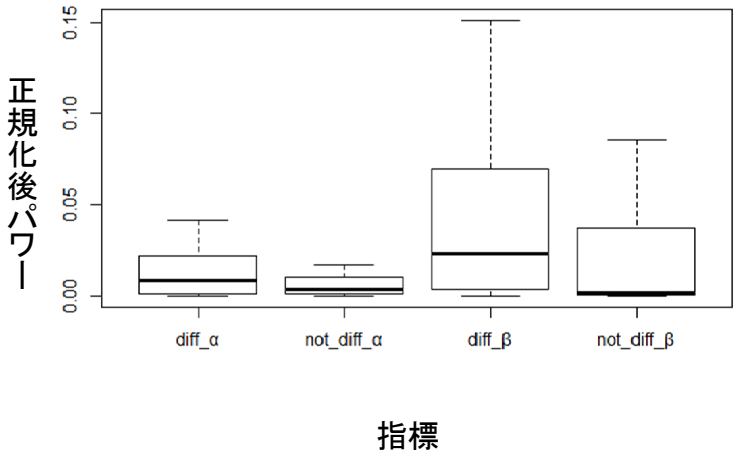

図 5.4: バグ判断ステップ中の脳波の比較(時間正規化後)

る．diffでは，中央値が0.008，平均値が0.025であった．not_diffでは中央値が0.003，平均値が0.011であった．つまり，バグ判断ステップ中のtnorm_αを比べると，中央値，平均値共にundの方がtnorm_αが大きかった．tnorm_βも同様に比べる．diffでは，中央値が0.022，平均値が0.050であった．not_diffでは中央値が0.001，平均値が0.024であった．つまり，バグ判断ステップ中のtnorm_βを比べると，中央値，平均値共にdiffの方がtnorm_βが大きかった．

以上の結果が偶然に起こったものでないか確かめるために検定を行った．データの等分散性はないものとして，Welchのt検定を行った．diff，not_diff間のtnorm_α

についてt検定を行うとp値は0.065．tnorm_βについてt検定を行うとp値は0.044となり，tnorm_βについて有意差が見られた．

以上の結果からプログラマが正しくバグ判断できる時は，正しくバグ判断できたない時よりもβ波のパワースペクトルが大きくなることが示された．この結果は，バグを正しく判断できなかった時にβ波が優位になるという予想に反している．この原因について考える．

β波は注意力が高まった時にも優位になる．バグかどうか判断するには，仕様とプログラムの動作が一致しているか判断しなければならない．このためには仕様，プログラム中の変数に格納される値，プログラム中のデータフロー等，多くの項目を確認する必要がある．正しく判断できる時には，多くの項目を確認しているため注意力が高まると考えられる．

反対に正しく判断できない時には，確認する項目に見落としがあり結果正しく判断している時よりも確認する項目が少なくなると考える．例えば，プログラム中のデータフローを理解していない場合は，データフローを気にせずに判断するはずである．確認する項目が少なくなれば，注意を払う対象が少なくなる．結果，正しく判断できない時には，正しく判断できる時に比べて注意力が低いのではないかと考えた．このような違いがβ波に反映されたと考えられる．この仮説を検証するためには，バグを正しく判断できる時には，仕様の確認やプログラム動作を確認を多く行っているのか調べる必要がある．これを確かめるためには計測した視線移動データを分析することによって調べることができると考えている．

6 おわりに

本研究では，バグかどうかを判断するプログラマの状態としてプログラムの動作を理解した後バグかどうかを判断するという2つのステップがあると考えた．そこで，人の状態を定量的に分析する手法として用いられている脳波と視線を計測し，2つのステップについて分析した．

本稿ではこの2つのステップにおいて計測した脳波データを分析した．計測したα，βを計測指標としてプログラムの動作を理解できた時とそうでない時で差を分析した．同様に，バグを正しく判断できた時とそうでない時で差を分析した．計測した脳波を個人で正規化した結果に対し統計処理を行うと，どれも有意差は出なかった．これらの結果は各タスクの所要時間による影響が考えられたため，各タスクの所要時間で正規化を行った．その結果，プログラム動作理解ステップではαについてp値は0.021となり，有意差が見られた．バグ判断ステップではβについてp値は0.044となり，有意差が見られた．

結果としてプログラムの動作を理解できた時にαが大きくなることとバグを正しく判断できた時にβが大きくなることが分かった．プログラムの動作を理解できた時は，落ちついた精神状態にあるのでαが大きくなったと考えられる．しかしバグを正しく判断できた時は，βが大きくなった．β波は注意力が高まった時にも優位になる．バグを正しく判断できる時，仕様とプログラム動作が一致するか判断するため，仕様やプログラム中のデータフロー等の多くの事に注意を払うと考える．反対に正しく判断できない時，見落とし等から注意を払う箇所が少ないと考える．この差がβ波に現れていると考える．これを検証するために，バグを正しく判断できる時には，仕様の確認やプログラム動作を確認を多く行っているのか調べる必要があると考えている．これを行うには，計測した視線データを分析すれば調べることができると考えている．

今後の研究では，バグを判断する際のプログラマの状態をより明らかにするために本稿で明らかになったバグを正しく判断できた時にβが大きくなる理由をより深く考察したい．そのためにバグを正しく判断できた時とそうでない時の視線移動に違いを見つけたい．これを行う方法として，計測した視線データをバグを正しく判断できた時とそうでない時に分類し分析することを考えている．また，プログラム動作を理解しているプログラマの状態を明らかにするためにプログラム動作を理解している時に特徴的な視線移動はないか調べたい．そのためにプログラム動作を理解している時とそうでない時の視線移動に違いを見つけたい．これを行う方法として，計測した視線データをプログラム動作を理解している時とそうでない時に分類し分析することを考えている．

また，プログラム動作理解とバグ判断の2つのステップを組み合わせた分析も行うことが出来る．例えば，プログラム動作を理解していてバグを正しく判断で

きなかった場合とプログラム動作を理解していなくてバグを正しく判断できなかった場合とを比較することが出来る．この2つに差を見つけることが出来て2つを識別することが出来れば，バグを正しく判断できなかった人に対してプログラマの状態に合った支援を行うことが出来る．例えば，プログラム動作を理解していないプログラマにはプログラム動作を理解できるような支援を行う．プログラム動作を理解しているプログラマにはプログラム動作理解を前提とした支援を行うといったことが出来る．

謝辞

本研究を行う上で，上野准教授には研究や論文の執筆に関してのアドバイスを星の数ほど頂きました．深く感謝申し上げます．査読を担当して頂いた松尾教授には中間発表や査読でコメントを頂きました．厚く御礼申し上げます．最後に，奈良高専情報工学科5年生の16名の学生には被験者実験に参加して頂きました．本当に有難う御座いました．

参考文献

[1] J. Siegmund，C. Kanster，S. Apel，C. Parnin，A. Bethmann，T. Leich，G. Saake，A. Brechmann：”Understanding Understanding Source Code with Functional Magnetic Resonance Imaging”，Proceedings of the International Conference on Software Engineering (ICSE)，pp.378-389 (2014).

[2] T. Fritz，A. Begel， S. C. Mller，S. Yigit-Elliott，M. Zger：”Using Psycho-Physiological Measures to Assess Task Difficulty in Software Development”，Proceedings of the International Conference on Software Engineering (ICSE)，pp.402-413 (2014).

[3] 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).

[4] R. Stein，S. 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).

[5] T. Musha，Y. Terasaki，H. A. Haque and G. A. Ivanitsky :”Feature Extraction from EEGs Associated with Emotions”，Artificial Life and Robotics，Vol.1，pp.15-19 (1997).

[6] 上野秀剛，中村匡秀，門田暁人，松本健一:”プログラマの視線を用いたコードレビュー性能の要因分析”，ソフトウェア工学の基礎XIII，pp.103-112 (2006).

[7] 満倉靖恵:”脳はウソをつかない脳波で判るあなたの真実”:日本耳鼻咽喉科学会会報，Vol.118，No.4，pp.461-465 (2015).

[8] 中川尊雄，亀井靖高，上野秀剛，門田暁人，鵜林尚靖，松本健一:”脳活動に基づくプログラム理解の困難さ測定”，コンピュータソフトウェア，Vol.33，No.2，pp.78-89 (2016).

[9] 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).

[10] 村上郁也:”イラストレクチャー認知神経科学”:株式会社オーム社 (2013).

[11] 音茂龍司，辻貞敏:”よくわかる脳波判読第3版”:金原出版株式会社 (2015).

[12] 堀忠雄:”生理心理学-人間の行動を生理指標で測る”:培風館 (2008).

[13] 宮田洋，藤澤清，柿木昇治，山崎勝男:”新生理心理学-生理心理学の基礎 ”:北大路書房 (1998).

[14] T. Oohashi，E. Nishina，M. Honda，Y. Yonekura，Y. Fuwamoto，N. Kawai，T. Maekawa，S. Nakamura，H. Fukuyama，H. Shibasaki：”Inaudiblehigh-frequencySounds Affect Brain Activity: Hypersoniceffect”: Journal of Neurophysiology，Vol.83，No.6，pp.3548-3558 (2000).

[15] 山本愛子，上野秀剛:”プログラムの実装戦略推定時における脳波の時系列分析”，ソフトウェアエンジニアリングシンポジウム (2017).