視線追跡装置を用いたデバッグプロセスの分析

Analysis of Debugging Process by Using an Eye Tracking System

門田 暁人　高田 義広　鳥居 宏次

Akito Monden, Yoshihiro Takada and Koji Torii

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

1　はじめに

ソフトウェアを組み込んだシステムが, テストの途中やリリースの後に予期せぬ挙動を示した時には, ソフトウェア中に潜むバグを, 早急に発見し除去することが重要である. ところが, 個々のバグの発見や除去に要する時間には, 極めて大きなばらつきがある [1]. その原因としては, 対象となるソフトウェアの特徴, バグの特徴, 作業者の特徴, 作業のプロセスの特徴, 作業の環境の特徴などの多くの要因が考えられる. ただし, どのような要因がどの程度に影響しているかについては, 明らかにされていない. デバッグの効率を向上させるためには, デバッグのプロセスを詳細に分析して, 効率に影響する要因を明らかにする必要がある.

ところが, デバッグのプロセスを詳細に分析することは, 容易でない. デバッグ中の作業者の意図を外部から推定することが困難であり, しかも, 作業者の意図がわからなければ, その様子は, 作業者が 2, 3 種類の行動を適当に繰り返しているようにしか見えないからである. 例えば, テキストエディタの操作, プログラムの実行などの繰り返しである. 効率に影響する要因が調べられる程度に詳細にプロセスを分析するためには, 絶えず変化する作業者の意図を追跡する必要があると考えられる.

そこで, 我々は, 視線追跡装置や視聴覚機器などを併用して, バグを発見するプロセスに限定して, 作業者の意図を追跡する実験を行なっている. この実験は, プログラムの理解のプロセスを対象とした文献 [2] などの実験と似た方針に基づいているが, バグを発見するプロセスを対象としていることが異なる. 本稿では, 実験のために構築した観測システムと, これまでの実験の結果について報告する.

2　観測システム

構築した観測システムは, 図 1 のように 2 室に設置した. 1 室は, 被験者にデバッグを行なってもらう開発室である.

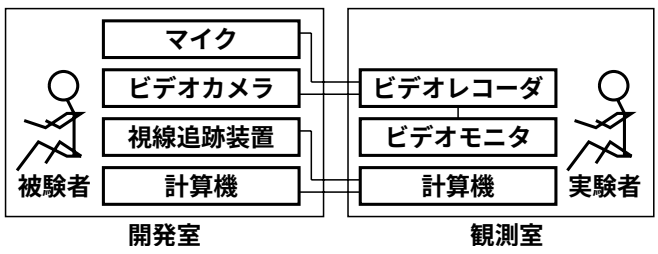

図 1: 構築した観測システム

図 2: 観測・分析用ツールの画面

開発室には, 防音を施して, 次の機器を設置した.

(1) 計算機. 被験者の作業用である. 開発したソフトウェアツールにより, ウィンドウやウィンドウ内の表示の変化と, キーストロークやマウスの操作のような入力とを, 観測室の計算機へ実時間で送信する.

(2) 視線追跡装置. 被験者の視ている画面上の点を測定する. これを画面上の注視点と呼ぶ. 被験者の頭に被ってもらうヘルメット型の部分と, 据え置き型の部分とからなる.

(3) マイク. プロトコル分析に用いる. つまり, 作業中に被験者に考えを絶えず話してもらい, それを録音する.

(4) ビデオカメラ. 被験者の表情や挙動を撮影する.

もう 1 室は, 実験者がデータを収集する観測室である. 次の機器を設置し, 観測室の機器とネットワークで接続した.

(1) 計算機. 開発室からのデータを受信し, 表示したり, 記録したり, 自由に再生したりできる. ウィンドウの表示の変化, 計算機への入力, 注視点の変化などを受信する.

(2) ビデオレコーダとモニタ. 被験者の映像や発話をモニタしたり, 記録したり, 再生したりできる.

システム中のハードウェアは全て市販のものであるが, ウィンドウの表示の変化, 計算機への入力, 注視点の変化を送受信, 表示, 記録, 再生するためのソフトウェアツールは, 新たに開発した. データを再生している様子を図 2 に示す. ある時刻において被験者が開いていた 2 個のウィンドウが再現されており, 瞳型の小アイコンによって注視点が表されている. コマ送り, 早送り, 巻戻しの機能も備えている. 現時点のシステムの注視点の測定精度は, 平均誤差が 15〜20mm になる程度である. つまり, ウィンドウの表示などに制約を設けなくても, 注視している文や変数などがおよそ識別できる.

.....	.....
13:19:43	(3 目並べゲームにおける) 次手を決定する関数があり, その中に 2 個の条件判定があると推測する. そして, それらの判定の順序が誤って入れ換わっていると言う仮説を立てる.
13:19:47	その箇所を探すために, 関数 main を頭から読み始める.
13:20:10	main では勝敗が表示されるだけであることを理解する. そして, 対局中の処理が関数 battle で行なわれていることを理解する.
13:20:11	関数 battle の記述を探しに行く. 瞬間的に発見する.
13:20:15	対局中の処理が battle で行なわれていることを確認するために, main の記述を読み直し始める.
.....	.....

図 3: プロセスの記述の例

3 実験

これまでに, 3 名の被験者に対して, のべ 5 回の試行を実施した. 各試行では, 1 名の被験者に, バグの症状だけを示して, そのバグを探してもらった.

3.1 各試行における手順

Step 0. バグの 1 個だけ入ったプログラムを用意する.
Step 1. 被験者にプログラムの仕様書を与え, また, 口頭でも仕様を説明する.
Step 2. 試行の度に視線追跡装置の調整が必要であるので, そのための 5〜15 分の作業を被験者に行なってもらう.
Step 3. プログラムを実行して, バグの症状を 1 個だけ被験者に見せて, 仕様と異なることを説明する.
Step 4. ソースプログラムを与えて, バグを探し始めてもらう. 考えを絶えず独り言のように話すよう要求する.
Step 5. デバッグを終了したと被験者が宣言するまで, 作業を続けてもらい, その間のデータを収集する.
Step 6. 記録したデータ (ウィンドウの表示の変化, 計算機への入力, 画面上の注視点の変化, 発話, 表情や挙動の映像) を一通り再生し, 被験者と一緒に作業を追体験する. そして, 各時刻に考えていたことを被験者に確認する.
Step 7. 被験者とは別に記録を繰り返し再生しながら, プロセスを非形式的に詳細に記述する.

3.2 プログラムとバグ

各試行では, 100〜402 行の C 言語のプログラムを対象とした. ごく短いプログラムを対象とした理由は, 分析の時間が非常に長くなることを避けるためである. 具体的には, 3 目並べゲームや酒屋の在庫管理などのプログラムを対象とした. 潜んでいたバグは, 配列の添字の誤りや if 文の条件式の誤りなどである. 各試行において被験者がデバッグに要した時間は 15 分〜38 分であった. その後の分析に要した時間は, 2〜5 時間であった.

4 結果

4.1 観測システムの有効性

バグを発見するまでの被験者の意図の変化は, 図 3 に例示するように, どの試行についても詳しく記述できた. 記述の妥当性については, 被験者へ再確認した. このような意図の推定には, 主に, 注視点の変化と発話との記録が役立つことがわかった. ただし, どちらか一方が欠けても, このような推定が困難であったと思われた. 発話は, 解釈が容易であるが, 詳細さや正確さに欠ける. 一定の時間に話せる量には限りがあるし, 被験者は作業に集中すると黙ってしまうことも多い. 注視点の変化は, それだけでは解釈が困難な部分が見られたが, 詳細な情報が正確に得られる. 例えば, 特定の単語などに, 一瞬, 目が止まる様子なども容易にわかる. 注視点の変化と発話との記録の併用が有効であることがわかった.

4.2 デバッグプロセスの特徴

前述のように観測したプロセスを比較したところ, プロセスの間で次のような共通点が認められた.
(1) プロセスは, 主に, バグに関する仮説を立てること, バグに関する情報を得るためのプログラムの実行, プログラムを読んで理解することからなる.
(2) 大部分の時間において, 被験者は, バグについての 1 個の仮説を強く意識しており, プログラムを読んだり実行したりする行動は, その仮説に起因している.

そして, 次のような相違点が認められた.
(1) ある被験者は, バグについての仮説を持たずに, プログラムを読んだり実行したりすることがある.
(2) プログラムの理解の戦略は, 場合によって大きく異なる. しかも, プロセスの途中でも頻繁に変わる.
(3) 理解できない部分に遭遇した時, または, 仮説に矛盾を見つけた時の行動は, 場合によって大きく異なる.

これらの相違点は, バグの発見の効率に影響する要因を示唆していると考えられる.

5 おわりに

構築した観測システムにより, 短いプロセスについては, 図 3 のように被験者の意図を追跡できることがわかった. ただし, その作業には, プロセスの約 8 倍の時間を要することもわかった. ツールを改良したり新たに開発することが必要であると考えている.

観測したプロセスを比較することにより, 4.2 で述べたようなプロセスの特徴が確認された. ただし, 5 回の試行しか行なえていないために, 効率に影響する要因を明らかにする程の知見は得られていないし, 異なる条件の下で同じ特徴が現れるかどうかはわからない. 今後は, 本実験を継続して, 分析を深める予定である.

参考文献

[1] 高田, 鳥居: “プログラマのデバッグ能力をキーストロークから測定する方法”, 電子情報通信学会論文誌, **J77-D-I**, 9, 646 – 655 (1994).

[2] 吉川: “プログラム理解過程を調べた認知実験報告 (1)”, 人工知能学会研究会資料, SIG-IES-9202-7, 53 – 61 (1992).