7L-07

チャットボットを利用したJavaプログラムのデバッグ支援手法

村田 匠† 仲山 冬野† 橋浦 弘明†
日本工業大学†

1. はじめに

近年のソフトウェア開発においてネットワークを介したリアルタイムコミュニケーションを実現するチャットツールが盛んに用いられている．チャットボットとは，人間の代わりにチャットを行うソフトウェアを指し，ソフトウェア開発者の間では煩雑な作業を自動化し，開発者の生産性を向上させるような DevBot[1]と呼ばれるチャットボットの利用も進んでいる．

また，プログラムのデバッグ方法の 1 つにトレースを利用する方法がある．トレースとはプログラム実行中のイベントを逐次的に記録したものである．トレースに基づくデバッグでは，始めにプログラムのトレースを取得しながら故障(failure)の再現を行う．次に，取得したトレースを故障地点から逆向きに検分することによって，変数の変化や制御の流れを読み取り障害(fault)の箇所を特定する．

2. 研究目的

トレースに基づくデバッグはブレークポイントを使った方法と比較して，事前にプログラムの実行を中断させる場所や，追跡対象となる変数をあらかじめ決めておく必要がないという利点がある．一方で，トレースを検分して障害原因の特定を行うことはユーザにとって負担が大きい[2]．本研究は前述のような問題に対して，チャットボットを利用し，トレースの検分範囲を狭めることによってデバッグの効率化を行う手法を提案する．

3. 関連研究

トレースに基づくデバッグ手法については，これまでも様々なものが提案されている．例えば Schulz と Bockisch [3]は Eclipse Java Development Tools (JDT)を拡張し，逆戻りデバッグが可能な Redshell を開発している．しかしながら Redshell を用いたデバッグでは，対象となるプログラムの実行に先立ってブレークポイントを設定しておく必要あるため，開発者はあらかじめプログラムの内部構造を把握しておく必要がある．

A Method for Debugging Java Programs using Chatbots
Takumi Murata† Toya Nakayama† Hiroaki Hashiura†
Nippon Institute of Technology†

4. 提案手法

トレースに基づくデバッグを行う場合に大きな問題となるのは，取得されたトレースの量が膨大な場合である．このような問題が発生するのは，対象のプログラムの制御構造が複雑であったり，GUI を持っていたり，サーバなどで長時間実行される場合などが考えられる．その一方で，障害箇所を特定するために開発者が必要とするトレースは，ほとんどの場合，全体のほんの一部分だけである．また，前述のとおりトレースの検分はプログラムの故障地点から逆方向に遡及するように行う．したがって，トレースの量が膨大であっても，その中から故障地点に対応する部分を見つけることができれば，トレースに基づくデバッグの実施が可能である．

著者らはこれらの性質に着目し，チャットボットを用いてトレース中の故障地点に対応する箇所の特定を支援する手法を提案する．具体的には，開発者はデバッグ対象となるプログラムの実行中に故障と思われる挙動を発見した場合，チャットボットに対して特定のメッセージを送信する．これにより，開発者はトレースを検索する際に，メッセージの送信時間とメッセージ中の文字列からトレースを検索することが可能になる．

5. ツールの実装

本研究のツールは大きくチャットボット部とプログラムのトレースを取得部の 2 つから構成されている．

チャットボット部はチャットツールからボット宛の特定のメッセージを取得し，取得した時刻を保存する機能を持っている．

トレースを取得部は櫻井ら[4]が開発した Traceglasses を用いた．ただし，Traceglasses はトレース取得時のタイムスタンプを取得する機能を備えていない．加えて，対象プログラム実行中に取得中のトレースの内容を検索することもできない．そこで，本研究ではトレース取得時に情報としてタイムスタンプを取得するように

すると共に，トレースの検分時にチャットボット部で取得した情報を基にトレース内の検索を行う機能を実装した．

チャットボットとの連携は Traceglasses の Search タブにチャットボットから得られた情報を検索するためのボタンから行う(図 1)．開発者はプログラム実行時にチャットボットに送信しておいた文字列をクエリとして入力し，このボタンを押すことにより，トレース中の故障地点の候補の提示を受けることができる．

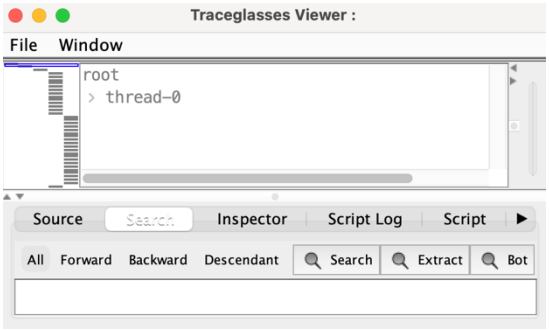

図 1: Traceglasses に対する拡張

6. 実験と考察

ツールの効果を確認するために実験を行った．被験者に対しては意図的に障害を混入したプログラムを配布し，これに対して提案ツールを用いたトレースに基づくデバッグを実施させる．実験に使用する問題はコマンドライン上で動作する 3 種類のボードゲームプログラムである．これらの問題に意図的に混入した障害は，プログラムの標準出力から発見できるものとした．

前述のとおり，障害箇所を特定するために開発者が必要とするトレースは，ほとんどの場合，全体のほんの一部分だけである．被験者がツールの支援によってトレース中から故障地点に対応する部分を発見し，適切にトレースを遡及することができれば，被験者は意図的に混入した障害に関係のないソースコードの内容を参照する必要はない．本研究では，被験者が障害に関係のないソースコードを参照した回数を誤ファイル参照数と呼び，ツールの有無によってその差が生じるかどうかを調査した．また，障害を特定するまでにかかる時間（トレース検索時間）についても調査した．

誤ファイル参照数とトレース検索時間をそれぞれ図 2, 3 に示す．これらの結果についてマン・ホイットニーの U 検定を行なった結果，誤ファイル参照数には有意差が認められなかった($\alpha = 0.05,\ p = 0.051$)一方，トレース検索時間に対しては有意差が認められた($\alpha = 0.05,\ p = 0.018$).

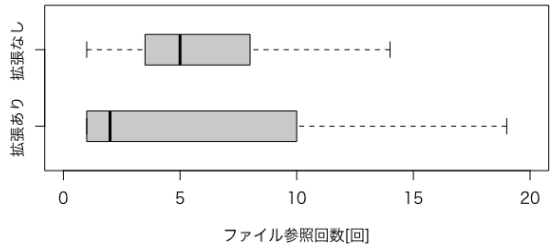

図 2: 誤ファイル参照数の比較

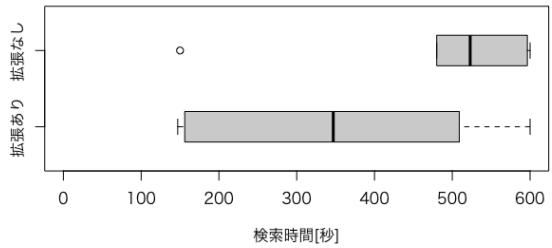

図 3: トレース検索時間の比較

7. 結論

本研究は，プログラム実行中に検索したい箇所をチャットボットに送信し，トレースを検索する手法を提案し，手法を実現するためのツールの実装を行った．評価実験により本手法はトレース検索時間の短縮に寄与することが明らかになった．

今後はさらに大規模なプログラムに対する適用や，ソフトウェア開発 PBL に対する適用などさらに実践的な環境での評価を行い，ツールの有効性を確認していきたい．

参考文献

[1] Linda Erlenhov, Francisco Gomes de Oliveira Neto, Philipp Leitner, “An Empirical Study of Bots in Software Development: Characteristics and Challenges from a Practitioner’s Perspective,” Proc. of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020), pp. 445-455, Nov. 2020.

[2] 松下 圭吾, 松本 真樹, 大野 和彦, 佐々木 敬泰, 近藤 利夫, 中島 浩, “実行トレースの比較を用いたデバッグ手法の提案及び評価,” 先進的計算基盤システムシンポジウム論文集, pp. 152-159, Mar. 2011.

[3] Stefan Schulz, Christoph Bockisch, “RedShell: Online Back-In-Time Debugging,” Proc. of Companion to the first International Conference on the Art, Science and Engineering of Programming (Programming ‘17), pp.1-2, Apr. 2017.

[4] 櫻井 孝平, 増原 英彦, 古宮 誠一, “Traceglasses: 欠陥の効率良い発見手法を実現するトレースに基づくデバッガ,” 情報処理学会論文誌プログラミング, 3 巻, 3 号, pp. 1-17, Jun. 2010.