JAWS 2013

モバイルエージェントのためのデバッグ環境の提案

尾崎 槙 Shin Osaki
鳥取大学大学院 工学研究科 情報エレクトロニクス専攻
Department of Information and Electronics, Graduate School of Engineering, Tottori University
s092018@ike.tottori-u.ac.jp

太田垣 真也 Shinya Otagaki （同　上）
s082009@ike.tottori-u.ac.jp

東野 正幸 Masayuki Higashino （同　上）
s032047@ike.tottori-u.ac.jp

高橋 健一 Kenichi Takahashi （同　上）
takahashi@ike.tottori-u.ac.jp

川村 尚生 Takao Kawamura （同　上）
kawamura@ike.tottori-u.ac.jp

菅原 一孔 Kazunori Sugahara （同　上）
sugahara@ike.tottori-u.ac.jp

Keywords: モバイルエージェント，デバッグ

Summary

モバイルエージェントシステムとはエージェントと呼ばれる自律的なプログラムがネットワーク上の計算機間を移動しながら動作するシステムである．モバイルエージェント技術はエージェントの移動という特徴により分散システムの設計において有効である．しかし，実社会においてモバイルエージェントシステムが普及しているとは言い難い．この原因の 1 つとして，エージェントの移動がモバイルエージェントシステムのデバッグを難しくしていることが挙げられる．モバイルエージェントシステムのデバッグでは，エージェントの移動により様々な問題が発生する．しかし，エージェントの移動に十分に対応したデバッガは提案されていない．そこで本研究では，モバイルエージェントシステムのデバッグの困難性を解決するために，エージェントの検索機能，システム内に存在するエージェントを一意に識別する機能，移動するエージェントに対するブレークポイント機能とステップ実行機能，及びエージェントとノードとの間の通信ログ管理機能を持ったモバイルエージェントシステムのためのリモートデバッガを開発する．エージェントの移動に対応したデバッガを開発することで，モバイルエージェントシステムのデバッグの困難性を解決する．

1. はじめに

モバイルエージェントシステムとはエージェントと呼ばれる自律的なプログラムがネットワーク上のノード間を移動しながら動作するシステムである．エージェントの移動やエージェント間のメッセージ交換といった特徴により，複雑な分散システムを人間が理解しやすい擬人的なモデルで構築可能となり，これまでに様々な応用システムが提案されてきた [Hurson 10][Outtagarts 09]．しかし，モバイルエージェント技術を用いたシステムが一般的に普及しているとは言い難い．その理由として，遠隔地のノード間を移動するエージェントのデバッグが難しいことが挙げられる．モバイルエージェントシステムのデバッグが難しい理由として，移動能力を持ったエージェントがデバッグ対象であること，多数のエージェントが多数のノード上で動作すること，エージェントがノードや他のエージェントと相互に影響を及ぼしながら動作することが主な原因として挙げられる．

モバイルエージェントシステムをデバッグするためには，多数のエージェントの中からデバッグの対象とするエージェントを見つける必要がある．しかし，多数のエージェントがノード間を移動しながら動作するため，どのエージェントをデバッグの対象にするかの判断や，その対象のエージェント見つけることが難しい．また，エージェントは遠隔地で動作するため，デバッグするためにはリモートデバッグ機能を用いる必要がある．しかし，一般的なリモートデバッグ機能は頻繁にリモート接続先を変更することを考慮しておらず，ノード間を移動を行うエージェントに対して利用することは難しい．さらに，エージェントはノード間を移動するため，様々なノードや他のエージェントか

らの影響を受けながら動作する．このため，発生した問題を再現することが難しい．

そこで本研究では，これらの問題を解決するために，エージェントの検索機能，システム内に存在するエージェントを一意に識別する機能，移動するエージェントに対するブレークポイント機能とステップ実行機能，及びエージェントとノード間の通信ログ管理機能を持ったモバイルエージェントシステムのためのリモートデバッガを開発する．これにより，モバイルエージェントシステムのデバッグの問題の解決を図る．

2. 関 連 研 究

システムを静的解析する手法の一つである形式手法をモバイルエージェントに応用した研究として，Mobile Object-Z (MobiOZ) [Taguchi 02, Taguchi 08], LAM (Logical Agent Mobility) [Xu 03, Ding 05]，π-ADL[Oquendo 04] では，モバイルエージェントの仕様記述を SPIN (Simple Process meta language INterpreter) モデルチェッカ [Ben-Ari 08] や CADP (Construction and Analysis of Distributed Proceses) toolbox [Garavel 11] といった検査器でモデル検査する手法を提案している．このような手法では，モデル検査により検証されたエージェントの仕様記述から，エージェントのプログラムコードを機械的に生成することで，実装上のバグを防ぐことができる．しかし，既存システムの開発で利用するには，専用の仕様記述言語を用いて開発するか，仕様記述言語と実装言語間の変換器を新規開発する必要があり，新たなコストを要する．また，システムが大規模で複雑な場合，形式手法ではモデル検査のための計算量が膨大になり適用が難しくなる．

動的解析する手法として，[Ndumu 99, Lam 05, Padgham 05, Vigueras 08, Cabac 09] では，エージェント間の協調関係を可視化するツールが提案されている．JADE [Bellifemine 10] や Agent Factory [Collier 07, Brazier 02] は，マルチエージェントシステムの開発・実行環境であり，エージェントの移動機能やエージェントのデバッガを提供している．しかし，これらはエージェント間のメッセージ交換に焦点が当てられており，移動を伴うエージェントの動作の解析には不向きである．[Lynch 07] ではマルチエージェントシステムの構築を支援する統合開発環境に必要となる機能について議論しているが，エージェントの移動性については今後の展望として簡単に言及されているだけである．[山谷 04] では，各ノードに残されたエージェントの移動履歴を追跡することでエージェントの現在地を特定し，ローカルでそのエージェントの動作履歴を表示できるエージェントビューワ MiSight を提案している．しかし，監視対象のエージェントを追跡するためには，エージェントの生成時から移動ログを各ノードに記録する必要があり，生成後のエージェントの位置特定やデバッグへの利用は難しい．

3. モバイルエージェントシステムのデバッグの困難性

モバイルエージェントシステムは複数のエージェントが多数のノードに分散してノード間を移動しながら動作する特徴を持つ．この特徴により，クライアントとサーバの 2 つのプログラムを準備する必要がないこと，耐障害性が高いこと，通信処理の簡素化ができることといった利点がある．しかしこの特徴は，以下に示す理由によりモバイルエージェントシステムのデバッグを難しくしている．

3·1 多数のエージェントが遠隔ノードで動作

モバイルエージェントシステムにおいて，デバッグの対象は主にエージェントであり，システム内には多数のエージェントが存在する．分散システム内に存在する多数のエージェントは遠隔地のノードへ移動して自律的に動作しているため，全てのエージェントを把握しておくことは難しい．このため，デバッグを実施するには，システム内から任意のエージェントを見つけ出し，どのエージェントをデバッグする必要があるのかを判断する必要がある．また，同じプログラムコードから生成された複数のエージェントが，自身のプログラムコードの組み替えによって，その後の動作を変える場合もある．このため，問題のある動作を行うエージェントをデバッグするには，システム内に複数存在するエージェントからデバッグの対象とするエージェントを一意に識別して捕捉する必要がある．

3·2 遠隔ノード間を移動しながら動作

モバイルエージェントシステムでは，多数のエージェントが遠隔地のノードを移動しながら動作する．このため，エージェントの動作を確認するには，エージェントが動作しているノードにリモート接続する必要がある．また，エージェントはデバッグ中もノード間を移動するため，エージェントをデバッグするためにはエージェントの移動に合わせてリモート接続先を変更する必要がある．しかし，一般的なリモートデバッグ機能ではリモート接続先を頻繁に変更することを想定していないため，デバッグ中に頻繁に移動するエージェントをデバッグすることは困難となる．このため，エージェントの移動を想定したリモートデバッガが必要となる．

3·3 ノードや他のエージェントからの影響

モバイルエージェントはノードを移動しながら動作するという特徴を持つ．このため，同じ処理であっても滞在するノードや他のエージェントからの影響によりエラーが発生する場合としない場合がある．このため，エラーを再現するにはエージェントの動作だけでなくエージェントがノードや他のエージェントから受ける影響も再現することが必要となる．

4. モバイルエージェントシステムのためのデバッガ

前述したモバイルエージェントシステムのデバッグの困難性の原因を取り除くために，モバイルエージェントシステムのための，エージェントの検索機能，システム内に存在するエージェントを一意に識別する機能，ブレークポイント機能とステップ実行機能，及びエラーの再現機能を開発する．

4·1 エージェントの検索

移動するエージェントの位置を検索する技術ではエージェントが持つ一意的な識別子を検索キーとして利用する方法が多数提案されている [小林 05]．しかし，デバッグにおけるエージェントの検索では，どのエージェントにバグが存在しているかは，あらかじめ分かるものではないため，一意的な識別子による検索だけでは不十分である．このため，システム内に多数存在するエージェントの中から任意のエージェントを見つけ出し，どのエージェントをデバッグする必要があるのか判断するために，様々な条件でエージェントを検索する機能を提供する．

エージェントを検索するためには，エージェントの名前，エージェントの状態，エージェントの場所などの属性値が検索条件として有効であると考えられる．一般的にエージェントの名前にはそのエージェントの役割が反映されている．このため，エラーが発生した処理担当に関わるエージェントを検索することで，バグを含む可能性が高いエージェントを見つけ出すことができる．エージェントの状態とは各エージェントの実行状態，待機状態，終了状態などの動作の状態を表すものである．状態による検索をすることで，開発者が意図していないエージェントが待機状態や終了状態へ遷移していることを発見できる．エージェントの場所とは，エージェントが動作しているノードの識別子である．エージェントは動作するノードからの影響でエラーを発生させる可能性があるため，場所による検索をすることでこれを発見できる．

また，システム上には同じ種類のエージェントが多数存在するため，デバッグ対象として指定するためにはこれらのエージェントを区別する必要がある．そこで，検索により得られた各エージェントに対して一意識別子を付与する．

4·2 ブレークポイント機能の移動対応

一般的にはブレークポイントはデバッグ対象のプログラムのソースコードに対して設定する．モバイルエージェントシステムでは，デバッグ対象となるプログラムはエージェントとして多数存在し，これらのエージェントの中には同じソースコードから生成されたエージェントや他のエージェントから複製されたエージェントなどが存在する．このため，1 つのソースコードから異常な動作をするエージェントが複数生成される可能性がある．しかし，エージェントは動作する環境からの影響を受けて動作するため，同じソースコードから生成されても実際に異常な動作をするのは一部のエージェントである場合が多い．このような場合，エージェント毎にデバッグができることが必要となる．そこで，ブレークポイントは各エージェントのソースコードに加えてエージェントの属性値または一意識別子の組に対して設定する．

エージェントをブレークする際は，まずエージェントの検索機能により見つけ出したエージェントの中からデバッグするエージェントを指定し，そのエージェントをデバッグモードへ移行させる．デバッグモード中のエージェントはデバッグを行うノードとの接続を維持する．次に，デバッグを行うノードでは，デバッグモード中のエージェントのソースコードにブレークポイントを設定し，エージェントに送信する．また，デバッグ中のエージェントが別のノードへ移動した後も移動前のデバッグを継続する必要がある．そのためには，設定したブレークポイントや移動回数のカウンターなどのデバッグに必要なデータを移動後もエージェントが保持しておく必要がある．そこで，デバッグ中のエージェントが移動する際には，エージェントのプログラムコードや実行状態に加えて設定したブレークポイントを移動することでデバッグを継続する．

i. ステップ実行機能

エージェントに対してステップ実行を行う場合，ステップ実行の間にノード間を移動する場合がある．その場合，ステップ実行の前後でリモート接続先を切り替える必要がある．デバッグ中エージェントは移動先を通知することで位置を把握できるため，その通知を元にリモート接続先を変更可能である．

ステップ実行には 1 つ先の命令で止めるステップインと 1 つ先の関数が完了した時点で止めるステップオーバーが存在する．この関数の中で，エージェントが短い間隔で連続した移動を行う場合は問題が発生する．なぜならエージェントが短い間隔で移動を行った場合は移動先の通知が複数送信され，これらの通知はそれぞ

れが別の通信で行われるため，通知が間違った順序で届く可能性がある．これにより，エージェントの位置を誤って把握してしまう可能性がある．デバッグを継続するためにはどの通知が最新のものであるかを判断する必要がある．そこで，この問題に対してはデバッグ中のエージェントに移動回数のカウンターを保持させることで解決する．エージェントの移動時には移動先のノードと移動回数のカウンターをデバッガに通知することで判断する．

これにより，移動をまたいだステップ実行を可能とし，エージェントに対応したステップ実行を実現する．

デバッグを行う場合，ブレークポイントの設定やステップ実行などのデバッグ処理を行う度にエージェントにアクセスする必要がある．エージェントはノード間を移動するため，その度に再検索する必要がある．しかし，これでは手間がかかる上に頻繁に移動するエージェントを捕捉できない．そこで，各ノードがデバッグ中のエージェントの移動を監視し，エージェントが移動した時に移動先のノードをデバッガに通知することでデバッガからエージェントの位置を把握する．これにより，デバッグ中のエージェントはデバッガ側から位置を常に把握する．

4·3 エラーの再現機能

エージェントは動作しているノードからの影響を受けて動作するため，エラーの発生はエージェントだけではなくノードの状態も関係する．このため，エラーを再現するにはエージェントの状態に加えて，ノードからの影響も再現する必要がある．エラー発生時の状態を完全に再現するためにはエージェントの起動から終了までの処理を全て再現する必要がある．そのためにはデバッグするエージェントの移動やノードからの影響を全て記録する必要があるが，現実的には不可能である．そこで，エラーが発生した時のエージェントの状態をスナップショットで記録し，さらに，デバッグ中のエージェントとノード間の通信を記録することでエラーの再現を支援する．

エージェントはプログラムカウンタやコールスタックなどの実行状態を内部に持つ．このため，エージェントの複製を記録することで，その時のエージェントの状態を再現することができる．そして，エージェントとノード間の通信を記録することでエージェントが受けた影響を再現することができる．

4·4 実　　装

デバッグ機能を我々が開発するモバイルエージェントフレームワーク Maglog[Kawamura 05] に実装した．Maglog におけるエージェントは Prolog Café[Banbara 05] を用いて実装されている．Prolog Café は Java で実装された Prolog から Java へのソースコード変換器及び Prolog インタプリタである．エージェントは Prolog の実行エンジンである Prolog クラスを継承した Agent クラスのインスタンスである．そして，このインスタンス毎に一意な識別子として UUID を付与する．

開発者はエージェントの振る舞いを Prolog で記述し，その内容を Java の述語クラスへ変換する．そして，各エージェント毎のスレッド上で述語クラスの読み込みと評価を繰り返すことでエージェントは動作する．よって，ブレークポイントは Prolog のソースコードに対して設定し，設定したブレークポイントの直前でエージェントのスレッドを停止することでブレークポイントを実現する．そして，停止したエージェントのインスタンスに対してリフレクション API を用いることでデバッグに必要なデータを取得し，そのデータをデバッガに送信することでリモートデバッグを行う．

また，エラーの再現機能におけるエージェント間通信について，Maglog では，エージェントとノード間通信は組み込み述語でのみ行われる．そこで，この述語内でデバッグ中エージェントの通信を記録することでエージェントとノード間の通信を記録する．そして，エージェントは Agent クラスのインスタンスであるため，Java の Object Serialization[Oracle Corporation 13] によって複製が可能である．

5. 評価方法の検討

一般的に行われているデバッグ手順は大きく分けて以下の 3 つの手順で行われる．そして，この手順をモバイルエージェントシステムに用いる場合，モバイルエージェントシステムの特徴によりそれぞれの手順で次のような問題が発生する．

(1) エラーの存在の認識
  (a) エージェントは遠隔地のノードで自律的に動作するため，各エージェントや各ノードの状態の把握が難しく，また，複数のノード間を移動しながら動作するため，広範囲の状態を把握する必要があり更に難しくなる．
(2) エラー発生原因の特定
  (a) エージェントをデバッグするには対象となるエージェントを捕捉する必要があるが，多数のエージェントが遠隔地に分散して自律的に移動しながら動作するため，エージェントの位置を把握することが難しく捕捉が難しい．
  (b) エージェントは遠隔地で動作するため，リモートデバッグ機能によりデバッグを行う．しかし，一般的なリモートデバッグ機能では頻繁な接続先の変更を考慮していないため，エージェントの移動に対応できない．
  (c) エージェントはノード間を移動しながら動

作するため，動作するノードや他のエージェントからの影響を受けて同じ処理でも異なる動作をする場合がある．これにより，特定のノードでのみエラーを発生させることもあるため，エラー発生時の状態を再現することが難しく，エラーの再現が難しい．

(3) 修正・動作確認

(a) 修正を適用するにはシステムを再起動する必要があるが，ノード数が増えれば再起動のコストが大きくなり動作確認が手間となる．

提案するリモートデバッガの評価として，このデバッグ手順で発生する困難性をどの程度軽減できるか評価することを考えている．

6. おわりに

本稿ではモバイルエージェントシステムの特徴により発生するデバッグの問題点を挙げ，それを解決するためにモバイルエージェントの移動を考慮したデバッグ機能を提案した．今後はデバッグ機能の改善とその他の問題点の解決を行い，このリモートデバッガの評価を行ないたい．

◇参考文献◇

[Banbara 05] Banbara, M., Tamura, N., and Inoue, K.: Prolog Cafe : A Prolog to Java Translator System, in *Proceedings of the 16th International Conference on Applications of Declarative Programming and Knowledge Management*, pp. 1–11 (2005)

[Bellifemine 10] Bellifemine, F., Caire, G., Trucco, T., and Rimassa, G.: JADE PROGRAMMER ' S GUIDE (2010)

[Ben-Ari 08] Ben-Ari, M.: *Principles of the Spin Model Checker*, Springer London (2008)

[Brazier 02] Brazier, F. M. T., Overeinder, B. J., Steen, van M., and Wijngaards, N. J. E.: Agent factory: generative migration of mobile agents in heterogeneous environments, in *Proceedings of the 2002 ACM symposium on Applied computing*, SAC '02, pp. 101–106, New York, NY, USA (2002), ACM

[Cabac 09] Cabac, L., Dörges, T., Duvigneau, M., and Moldt, D.: Requirements and tools for the debugging of multi-agent systems, in *Proceedings of the 7th German conference on Multiagent system technologies*, MATES'09, pp. 238–247, Berlin, Heidelberg (2009), Springer-Verlag

[Collier 07] Collier, R.: Debugging agents in agent factory, in *Proceedings of the 4th international conference on Programming multi-agent systems*, ProMAS'06, pp. 229–248, Berlin, Heidelberg (2007), Springer-Verlag

[Ding 05] Ding, J., Xu, D., He, X., and Deng, Y.: Modeling and Analyzing a Mobile Agent-based Clinical Information System, *The International Journal of Intelligent Control and Systems*, Vol. 10, No. 2, pp. 143–151 (2005)

[Garavel 11] Garavel, H., Lang, F., Mateescu, R., and Serwe, W.: CADP 2010: a toolbox for the construction and analysis of distributed processes, in *Proceedings of the 17th international conference on Tools and algorithms for the construction and analysis of systems: part of the joint European conferences on theory and practice of software*, TACAS'11/ETAPS'11, pp. 372–387, Berlin, Heidelberg (2011), Springer-Verlag

[Hurson 10] Hurson, A. R., Jean, E., Ongtang, M., Gao, X., Jiao, Y., and Potok, T. E.: Recent Rdvances in Mobile Agent-Oriented Applications, in Zomaya, A. Y. ed., *Mobile Intelligence: Mobile Computing and Computational Intelligence*, Wiley Series on Parallel and Distributed Computing, John Wiley & Sons, Inc., 1st edition (2010)

[Kawamura 05] Kawamura, T., Motomura, S., and Sugahara, K.: Implementation of a Logic-based Multi Agent Framework on Java Environment, in *Proceedings of IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pp. 486–491 (2005)

[Lam 05] Lam, D. N. and Barber, K. S.: Debugging agent behavior in an implemented agent system, in *Proceedings of the Second international conference on Programming Multi-Agent Systems*, ProMAS'04, pp. 104–125, Berlin, Heidelberg (2005), Springer-Verlag

[Lynch 07] Lynch, S. and Rajendran, K.: Breaking into industry: tool support for multiagent systems, in *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, AAMAS'07, pp. 136:1–136:3, New York, NY, USA (2007), ACM

[Ndumu 99] Ndumu, D. T., Nwana, H. S., Lee, L. C., and Collis, J. C.: Visualising and debugging distributed multi-agent systems, in *Proceedings of the third annual conference on Autonomous Agents*, AGENTS '99, pp. 326–333, New York, NY, USA (1999), ACM

[Oquendo 04] Oquendo, F.: π-ADL: an Architecture Description Language based on the Higher-Order Typed π-calculus for Specifying Dynamic and Mobile Software Architectures, *ACM Software Engineering Notes (SEN)*, Vol. 29, No. 3, pp. 1–14 (2004)

[Oracle Corporation 13] Oracle Corporation, : Object Serialization, Web (2013), `http://docs.oracle.com/javase/tutorial/jndi/objects/serial.html`

[Outtagarts 09] Outtagarts, A.: Mobile Agent-based Applications : a Survey, *International Journal of Computer Science and Network Security (IJCSNS)*, Vol. 9, No. 11, pp. 331–339 (2009)

[Padgham 05] Padgham, L., Winikoff, M., and Poutakidis, D.: Adding debugging support to the Prometheus methodology, *Eng. Appl. Artif. Intell.*, Vol. 18, No. 2, pp. 173–190 (2005)

[Taguchi 02] Taguchi, K. and Song Dong, J.: An Overview of Mobile Object-Z, in *Proceedings of the 4th International Conference on Formal Engineering Methods: Formal Methods and Software Engineering (ICFEM-2002)*, pp. 144–155 (2002)

[Taguchi 08] Taguchi, K. and Song Dong, J.: Formally specifying and verifying mobile agents model checking mobility: the MobiOZ approach, *International Journal of Agent-Oriented Software Engineering*, Vol. 2, No. 4, pp. 449–474 (2008)

[Vigueras 08] Vigueras, G. and Botia, J. A.: Tracking causality by visualization of multi-agent interactions using causality graphs, in *Proceedings of the 5th international conference on Programming multi-agent systems*, ProMAS'07, pp. 190–204, Berlin, Heidelberg (2008), Springer-Verlag

[Xu 03] Xu, D., Yin, J., Deng, Y., and Ding, J.: A Formal Architectural Model for Logical Agent Mobility, *IEEE Transactions on Software Engineering*, Vol. 29, No. 1, pp. 31–45 (2003)

[山谷 04] 山谷 孝史, 服部 宏充, 伊藤 孝行, 新谷 虎松：モバイルエージェントのための位置透過な操作手法とエージェント分散処理への応用, 人工知能学会全国大会 (第 16 回) 論文集 (2004)

[小林 05] 小林 晋輔, 石井 雅将, 小野里 好邦, 河西 憲一：モバイルエージェントの動きを利用した位置管理, 情報処理学会論文誌, Vol. 46, No. 2, pp. 506–516 (2005)