モバイルエージェントの移動性を考慮したデバッグ手法の提案

尾崎 槙[†] *，東野 正幸[‡]，高橋 健一[†]，川村 尚生[†]，菅原 一孔[†]
([†] 鳥取大学大学院 工学研究科，[‡] 鳥取大学 産学・地域連携推進機構)

A Debugging Method Taking Account of Mobile Agent's Mobility

Shin Osaki[†] *, Masayuki Higashino[‡], Kenichi Takahashi[†] , Takao Kawamura[†] , Kazunori Sugahara[†]
([†]Graduate School of Engineering, Tottori University
[‡]Organization for Regional Industrial Academic Cooperation)

1. はじめに

モバイルエージェントシステムとは，モバイルエージェントと呼ばれる自律的なプログラムが，ネットワークに接続されたノード間を移動しながら問題を解決していくシステムである．エージェントの移動やエージェント同士の協調といった特徴は，分散システムの構築技術として期待されている．しかし一方で，これらの特徴がデバッグを難しくする要因となっている．これに対して，エージェントの協調関係や動作シナリオを定義するインタラクション記述言語による手法や[(1)]，エージェントのインタラクションを視覚化することでデバッグを行うツール[(2)]などが提案されている．しかし，いずれもエージェントの移動を十分に考慮していない．

そこで本論文では移動するエージェントをデバッグするために，バグに関わるエージェントを発見するためのエージェント検索機能，移動するエージェントを追従するために動的に接続先を変更可能なステップ実行機能，エージェントの動作の記録によるバグの再現支援機能を提案する．これにより，モバイルエージェントシステムのデバッグにおける困難性の軽減を図る．

2. モバイルエージェントシステムのデバッグの問題

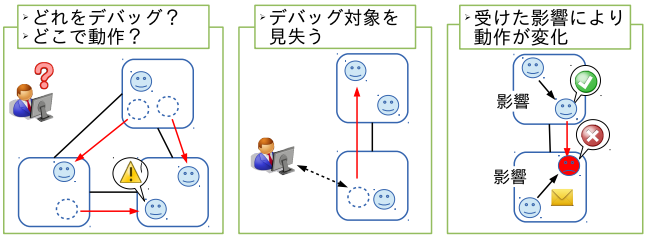
エージェントの移動や協調動作によって，デバッグ時に以下のような問題が発生する．

問題 1. 自律的な移動

モバイルエージェントシステムにおけるデバッグの対象はエージェントである．エージェントはシステム内に多数存在し，それぞれが自律的に動作すると共にノード間を移動するといった特徴を持つ（図 1(a)）．この特徴により，各エージェントの動作を把握することが難しくなり，バグの原因となるエージェントの発見が困難となる．このため，開発者がバグの原因となっているエージェントを見つけ出す仕組みが必要となる．

問題 2. デバッグ中の移動

遠隔ノードで動作しているプログラムの動作は，そのプログラムが動作しているノードにリモート接続して確認する必要がある．しかし，リモート接続先とは異なるノードへエージェントが移動した場合，エージェントを見失う（図 1(b)）．このため，エージェントの動作内容を継続的に確認するために，エージェントの移動にあわせてリモート接続先を変更する仕組みが必要となる．



(a) 自律的な移動　(b) デバッグ中の移動　(c) 他のエージェントからの影響

図 1. モバイルエージェントシステムのデバッグの問題

問題 3. 他のエージェントからの影響

バグの原因を確定するためには，そのバグを再現する必要がある．エージェントは他のエージェントや滞在しているノードといった周囲の状況に応じて動作が変化し，その変化もまた周囲の状況へ影響を及ぼす．さらにその影響はエージェントの移動により複数のノードへ拡散する．さらに，状況の変化に関わったノードがネットワークから離脱する可能性もあり，モバイルエージェントシステムにおけるバグの再現は難しい（図 1(c)）．このため，エージェントの動作内容だけでなくエージェントが受けた影響を記録・収集し，バグの再現を支援する仕組みが必要となる．

3. モバイルエージェントシステムのためのデバッグ機能

〈3·1〉 エージェントの検索機能　デバッグするためにはまず，バグを持つエージェントを見つけ出す必要がある．バグの可能性のあるエージェントは開発者が想定しないノードに移動したり，動き続けるはずが長時間待機したり異常停止したりする場合がある．そこで，これらの異常な動作を発見するために，各ノードでエージェントを監視

し，エージェントの移動ログとエージェントの状態を記録する．システムの動作がおかしい時に，その動作に関連すると思われるエージェントを，移動したことのあるノードやエージェントの動作状態で検索することで，バグの可能性のあるエージェントを絞り込む．

〈3･2〉 エージェントに対するステップ実行機能 一般的なプログラムのデバッグにはステップ実行機能を用いる．これをモバイルエージェントシステムに対して用いる場合は，継続的にデバッグするために遠隔地のノード間を移動するエージェントの移動に合わせてリモート接続先を切り替える必要がある．これは，エージェントの移動先をデバッガに通知し，その通知を元にリモート接続先を変更することで対応する．しかし，ノード毎に通信速度が異なるため，各ノードで送信された順序と実際に通知が届く順序が異なる可能性がある．そこで，エージェントの最新の位置を把握するために，デバッグ中のエージェントの移動回数を記録し，移動時に共に通知する．そして，移動回数から順序を判断することでエージェントの位置を正しく判断し，リモート接続先を変更する．

〈3･3〉 バグの再現支援機能 モバイルエージェントシステムではバグの再現が難しい．そこで，エージェント毎の動作を記録し，バグ発生時に記録したエージェントの状態を復元することで，バグの原因特定を支援する．

過去のエージェントの動作を再現するためにはエージェントの状態を記録する必要がある．ここで，エージェントは実行時状態を内部に保持し，ノード間の移動時に自身の複製を生成する．そこで，エージェントの移動時に複製を記録する．また，エージェントは他のエージェントとメッセージ交換し，そこから影響を受けて動作する．このため，このメッセージの内容，どのエージェントから受け取ったか，どのノードで受け取ったか等を記録する．これらの情報を各ノードにログとして保存し，開発者が異常を感じた際に検索機能を用いて収集する．開発者の元で仮想的なエージェント実行環境を構築し，その上でエージェントの複製から過去の状態を復元し，メッセージ交換を再現する．これにより，バグの再現を支援する．

4. 評　価

提案するデバッグ機能をアプリケーション開発時に利用することで提案機能の有効性を検証した．アプリケーションとしては，分散ハッシュテーブルアプリケーション (DHT) とホテル予約アプリケーション (HOTEL) を作成した．これらの開発時に発生したバグとその解決に有効に働いた機能を表 1 に示す．例えば DHT アプリケーションでは，データの取得後にデータ取得エージェントがアドレスの一覧から間違ったアドレスを取得したことでバグが発生した．

開発者はデータ取得エージェントが戻らないことからバグの存在に気づいた．そこで，エージェントがどこのノードで動作しているかを確認するために，検索機能を用いてデータ取得エージェントの位置を確認した．これにより，想定外のノードへ移動していることが分かった．そこで，バグの再現機能を用いて想定外のノードへ移動する直前の状態に復元し，ステップ実行機能を用いて移動処理を確認した．その結果，移動先のアドレスが間違っていること，また，そのアドレスを取得した一覧に正しいデータが無いことがわかった．これにより，バグの原因はアドレスを管理する別のエージェントにある事を特定した．

	発生したバグ	検索	ステップ実行	再現支援
DHT	データの保存に失敗	✓	✓	✓
	データの取得に失敗	✓	✓	✓
	ノードの参加に失敗		✓	✓
HOTEL	ホテル情報の準備に失敗		✓	
	情報の一部に欠損	✓	✓	
	検索情報とのミスマッチ	✓	✓	✓

表 1. 発生したバグ及び有効に働いたデバッグ機能

このデバッグで確認が必要だった情報は主にエージェントの位置，異常な動作の確認，具体的な変数の値であった．提案デバッガ無しではエージェントの位置を確認するために移動処理に関するログを出力し，各ノードで出力されるログをひとつひとつ確認する必要がある．また，異常な動作と具体的な変数の値のを確認ではリモートデバッグ機能を用いても，エージェントの移動によりエージェントを見失う．さらに，バグの発生後に手順を再現することが難しい．一方，提案デバッガを用いた場合は，検索機能によりエージェントの位置を，ステップ実行機能によりエージェントが移動したとしても変数の値や動作内容を容易に確認できた．さらに，再現支援機能により容易にバグを再現することができ，デバッグの困難性を削減できた．

デバッグに要したタイプ数とクリック数を表 2 に示す．タイプ数は 41%，クリック数は 24%削減でき，提案するデバッガはデバッグにかかる手間を削減できた．

	タイプ数	クリック数
デバッガ有	869	167
デバッガ無	514	128
削減率	41%	24%

表 2. デバッグ時のクリック数とタイプ数の削減率

5. おわりに

本稿ではモバイルエージェントシステムにおけるデバッグの問題点を示し，それら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．アプリケーション開発時に本提案機能を用いることで，モバイルエージェントシステムのデバッグを支援できる事を示した．

文　献

(1) 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).

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