B-026

モバイルエージェントシステム AgentSphere の開発
― デバッグ補助機能を持つオブジェクト操作可能なシェルの開発 ―
Development of Strong Migration Mobile Agent System AgentSphere
- Development of a Shell with Functions of Debugging and Operating Active Objects -

大久保 秀† 甲斐 宗徳‡
Shu Okubo Munenori Kai

1. はじめに

AgentSphere の開発段階では，その開発者が作成したエージェントやモジュールの振る舞いを確かめたり，動的にデバッグを行う必要がある．

例えば， JVM が大量のメモリを使用することで，JVM の処理能力が落ちるため，AgentSphere が動く JVM のメモリ使用率がある一定の値以内であるとき AgentSphere がエージェントを収容できる，という判断基準を定義したとする．しかし，実際にそのメモリ使用率となっても，JVM の処理能力が落ちなければ，判断基準の値を上げることができる．判断基準の値の変更を試行錯誤で何度も行うとき，そのたびに起動している AgentSphere を止めなければならないため，時間的ロスが発生する．また，エージェントは AgentSphere が停止する前にほかの AgentSphere に移動するという性質を持つので，AgentSphere を再起動させると，停止前に AgentSphere 内にいたエージェントはその AgentSphere 内からいなくなってしまう．エージェントは自律的に移動するため，AgentSphere 停止前の，エージェントが AgentSphere 内にいることによって JVM のメモリ使用率が判断基準の値に達しているという状況を再現するのは困難である．そのため AgentSphere を停止することなく，同じ状況のまま動的に判断基準としての値を入れ替える必要がある．

そこで，エラーを再現するのが困難な AgentSphere 内で，エラー自体から自律的に復旧するための変更を，エラーが起きた状態のまま動的にエージェントに行い，エージェントやモジュールの動作確認やデバッグをするため，動的なオブジェクト操作を可能にするシェルを開発した．

動的なオブジェクト操作とは，AgentSphere を動かしながら，エージェントやモジュールを構成するインスタンスのフィールドのデータ型やその中身の確認・変更，メソッドの引数や返り値の確認，メソッドの実行，メソッドの内容（コード）の変更を行うことである．

本研究によりメソッドの内容の変更以外のオブジェクト操作を行えるようになった．

2. 動的なオブジェクト操作の方法

動的にオブジェクトを操作するために Java の API であるリフレクション機能を用いた．リフレクションとはクラスのフィールドやメソッドの情報を取得，フィールドの値の変更，メソッド名の文字列からそのメソッドの実行を行うことができる Java の API である．

リフレクションの機能を呼び出すには，呼び出すための記述をコードに書く必要がある．従ってリフレクションの機能が呼び出されるタイミングは決まっている．そのタイミングを変更するにはコードを書きなおさなければならず，AgentSphere を停止する必要がある．そこで AgentSphere を止めることなくリフレクションの機能を任意のタイミングで対話的に呼び出し，動的なオブジェクト操作を行うためのインターフェイスとしてシェルを開発した．

2.1 動的なオブジェクト操作の流れ

今回開発したシェルは AgentSphere のモジュールの１つであり，AgentSphere と同時に起動する．そしてコマンドプロンプトからユーザの入力を受け取り，その内容を解釈して，ユーザからの命令に対応した処理を行う。この処理は図 2-1 に示すクラスを呼び出し，そのクラスの機能が働くことによって行われる．

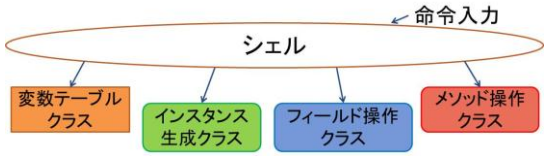

図 2-1 動的なオブジェクト操作の流れ

2.2 変数テーブルクラス

このシェルでは，シェルに

$文字列 = 値

と入力することで，”$”から始まる文字列にインスタンスの持つフィールドの値や，シェルで新たに生成したインスタンス，あるメソッドを実行した返り値を格納できる．シェルでこの変数を入力することによって，格納された値をいつでも参照し，値の利用や変更を行うことができる．

2.3 フィールド操作クラス

エージェントやモジュールのデバッグのために変数の持つ値を変更するには，まずその目的とする変数にアクセスする必要がある．フィールド操作クラスは変数へのアクセスとその変数の Read/Write を行うものである．

あるクラスのインスタンス変数またはクラス変数から，

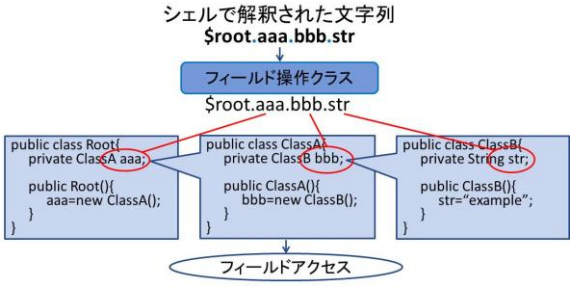

図 2-2 フィールドアクセスの流れ

† 成蹊大学理工学研究科理工学専攻 Graduate School of Science and Technology, Seikei University

変数をいくつかたどり，目的の変数にアクセスするには，それらの変数名を”.”でつなげてシェルに入力する．図 2-2 は Root クラスのインスタンス”$root”からたどった，ClassB のインスタンス変数”str”にアクセスする様子を示している．

こうしてアクセスした変数は，

$root.aaa.bbb.str = “change”

で中身を入れ替えたり，

$string = $root.aaa.bbb.str

でシェルで扱える変数に格納することができる．

2.4 メソッド操作クラス

エージェントやモジュールの動作を確かめるためには，動的にそれらのメソッドを実行する必要がある．メソッド操作クラスは指定したメソッドの実行を行うものである．

あるクラスのメソッドを実行するには，

インスタンス.メソッド名()

と入力する．このとき，引数がある場合は”()”内に入力する．これによりメソッド名と引数の型から，特定のメソッドが実行される．引数の型を指定することから，メソッド名が同じであっても目的のメソッドを実行することができる．ここで，引数として入力された文字列は String 型なので，図 2-3 に示すように，他のクラスの機能を呼び出し，目的とする型の値を取得する．

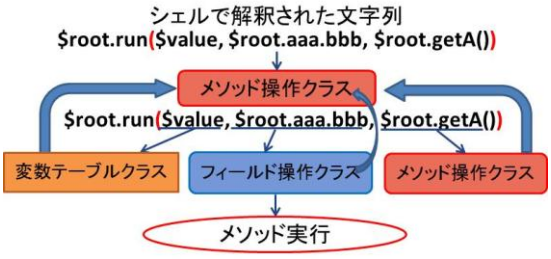

図 2-3　メソッド実行の流れ

2.5 インスタンス生成クラス

AgentSphere で新規のエージェントを動かすためには，エージェントの基底クラスのインスタンスを新たに生成し，エージェントが動作するためのメソッドを実行する必要がある．インスタンス生成クラスはシェルで新たなインスタンスの生成を行うものである．

新たなインスタンスを生成するには，

new パッケージ名.クラス名()

と入力する．メソッド実行のときと同様，引数は”()”内に入力することによって，特定のクラスのコンストラクタを呼び出され，インスタンスを生成する．生成されたインスタンスはシェル変数に格納するか，インスタンスの持つフィールドに代入しなければならない．

2.6 AgentSphere 内のオブジェクト操作

エージェントやモジュールのオブジェクト操作をするとき，それらが図 2-2 のサンプルコードのように複数のインスタンスで構成されている場合，まずその基底クラスのインスタンス（図 2-2 の場合は Root クラスのインスタンス）にアクセスする必要がある．

エージェントは起動時に，動作中のエージェントが追加されるクラスに追加される．したがって，エージェントの場合は，図 2-4 のように起動したエージェントが追加されていくクラスから，基底クラスのインスタンスにアクセスすることができる．モジュールの場合も，エージェントとは別のクラスに追加されていくが，同様の方法でアクセスすることができる．

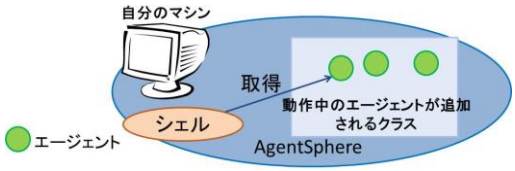

図 2-4　基底クラスのインスタンスへのアクセス

3. 動作実験

今回開発したシェルにより，動的なオブジェクト操作が可能となったことを確認するために次の実験を行った．

3.1 実験内容

テスト用のエージェントを作成し，そのエージェントのフィールドの値を変えることで，エージェントの振る舞いを変えるテストを行った．実験に使用したエージェントは指定した秒数ごとに現在時刻の出力を行う．このエージェントは変数として出力する時間の間隔を秒数で示す int 型の変数”interval”を持っている．”interval”には初期値として 5 が与えられており，初めは 5 秒ごとに出力を行う．

3.2 実行結果

初めにシェルで，実験に使用するエージェントのインスタンスを生成し，シェルで使用される，任意で作成した変数”$test”に格納する．次に生成したインスタンスの持つ，エージェントが動き出すためのメソッドを実行させる命令をシェルに入力する．これによりエージェントが 5 秒ごとに出力を行う．そしてシェルに”interval”の値を変更するための記述を入力する．例えば，” $test.interval=10”と入力すると，10 秒ごとに出力されるようになった．上の 10 の部分を-1 として入力すると，エラーを起こして処理が止まった．再び”interval”の値を 5 にする入力をし，エージェントが動き出すためのメソッドを実行させる命令の入力をすると，エージェントは再び動き始め，5 秒ごとに出力を行った．これによって，フィールドの持つ値を変えることで，エージェントの振る舞いを変えることができた．

4. おわりに

今回開発したシェルによって，AgentSphere を構成するすべての要素を動的にオブジェクト操作できるようになり，エージェントやモジュールの振る舞いを確認したり，エラーが起こったときのデバッグを行うことができるようになった．これにより，今後の AgentSphere 開発の効率が上がると思われる．今後の課題として，メソッドの変更を行うための実装をする必要がある．

謝辞

本研究は科研費（基盤研究(C)21500041）の助成を受けたものであることをここに記し、謝意を表します。

参考文献

[1] 鈴木幸祐・山口大祐・甲斐宗徳「モバイルエージェントシステム AgentSphere における強マイグレーション機構の改良」,FIT2011, B-030,Sep.2011.