卒業論文　マシンエージェント作成入門編

02 c 2126003F

有賀ゼミ　会計学科 4 年阿部光良

目次

1 序文

当初、このゼミに入学したときは、パソコンを使ったゼミであり前々からパソコンを用いて様々な事をやってきた私にとっては親しみやすいものであろう、といった簡単な動機からこのゼミを選んだ。最初は自分でも簡単にできると思っていたのだが、勉強を進めるにつれて、自分がいままで触れる機会の少なかったプログラミング、経済学の分野など幅広い知識が必要となり、自分もこの卒業論文を書く程度にいたるまでは苦労を強いられた。しかし、こうして苦労したことは自分の知的刺激になったと本当に思えるし、苦労してよかったと感じる。

さて、この論文の中心となるものは、マシンエージェントの作成方法である。

先輩方の論文や、インターネットを使って調べてみたのだが、どれも初学者には厳しいものばかりであったと感じた。プログラムの断片だけインターネット上にある状態のサイトもあり、中心となるマシンエージェント本体を、最初私は見つける事ができなかった。実際、後輩にマシンエージェントに関して質問したところ「独学でやった。最初は先輩方のプログラムをそのまま使ってみたりもした」といった意見を聞いた。これを聞いて、私はマシンエージェントを作るには一定の基礎知識がある事を前提に進められている、と感じた。

それゆえ私は「初心者でも、努力すればマシンエージェントを作る事ができるような状態にするべきではないか」と思いこの論文を作成しようと思いました。プログラミングの分野は、パソコンに慣れ親しんでない人や英語が苦手な人には拒否反応すら出る場合もある。そういった方でも私の論文を読んでプログラミングに親しみを持ち、そして自分のオリジナルのマシンエージェントが簡単に作ることができるようになれれば、本望である。そして、これからの有賀ゼミを支え、発展させるのに貢献して欲しい。

2　U-mart に関して

U-mart とは仮想の取引市場を構築できるプログラムの事である。既に諸先輩方の論文にもこの項目はあるので、出来るだけ省いて説明したい。
一応、公式的な見解としては、インターネットを介して離れた場所からでも対戦が可能である、マシンエージェントを組み込み、自分なりの戦略を立てることが出来るなどある。
また、U-mart は経済学のみならず、心理学、工学の分野からも注目されているプログラムでもある。

しかし U-mart をやっていて感じたことが 1 つある。
それは、どのマシンエージェントも単体で行動するもののみである。U-mart は先物商品のみ取引可能なのでそれも仕方のない事であるとは思うが、もしマシンエージェントが複数で互いに連携して行動するとさらに利益があがるようなシステムが形成されているのであれば、つまり、先物のみの取引ではなくほかの要素も含めた U-mart になれば、さらに研究の余地が広く、そして深く出来るのではないだろうか、と思った。

しかし、そのように U-mart 自体が巨大化した場合にも問題がある。それはマシンエージェント作成がさらに困難なものになり、複雑になってしまう事だ。現在の状況でさえ、私を含め多くのゼミ生がマシンエージェントを完全に独自の手法で作れないような状況では、たとえ U-mart に他の要素が加わったとしても、ゼミ生がついていけなくなり、一部の有能なゼミ生のみで研究が進んでしまう状態も考えられる。

3-1 マシンエージェント作成に関して

ここで、この論文の中心となるマシンエージェントに関しての話をしたいと思う。まず、マシンエージェントを作ろうと思ったら、UMIE と eclipse の入手をすべきである。ともに検索エンジンで検索すれば出てくる。そして、事前に java 環境が必須となるので、自宅のパソコンでやる場合においてはこれも各自敵似れておいて欲しい。そして、eclipse の起動には java 環境が必要なので各自ダウンロードして手に入れて欲しい。
やり方としてはまず、java 実行環境の入手から説明したい
http://java.sun.com/j2se/1.5.0/ja/download.html　に行き

J2EE 1.4 SDK　のダウンロードをクリックして解凍し、インストールすれば java 環境は出来る。
UMIE は http://www.u-mart.org/html/u-mart-kit/　にある

UNMIE2005参加用U-Mart開発キット(2005年8月10日バージョン)umie2005sdk050810.zip

をクリックすることにより入手できる。
最後に eclipse に関してである。
Eclipse はただダウンロードしてもすべてが英語であるのでまずは日本語化キットを入手した方が、研究が順調に進むと思われるので、そのやり方を以下に書き記して行きたい。
まずは http://download.eclipse.org/eclipse/downloads/index.php　に行き

Language Pack　　3.1.1 Language Packs

をクリックする。
そして OS が Windows なら、

SDK Language Packs	Windows 98/ME/2000/XP	Linux (x86/GTK 2)
NLpack1 - German, Spanish, French, Italian, Japanese, Korean, Portuguese (Brazil), Traditional Chinese and Simplified Chinese.	NLpack1_FeatureOverlay-eclipse-SDK-3.1.1.zip NLpack1-eclipse-SDK-3.1.1a-win32.zip	NLpack1_FeatureOverlay-eclipse-SDK-3.1.1.zip NLpack1-eclipse-SDK-3.1.1a-gtk.zip

下の Nlpack1-eclipse-SDK-3.1.1a-win32.zip をクリックして

これのどちらでも構わないのでダウンロードして保存する。次に最新版の eclipse をダウンロードして、上記の Nlpack1-eclipse-SDK-3.1.1a-win32 を eclipse 本体のある所へ貼り付けして、上書きしてもらえば、日本語で eclipse が使用できる。

3-2 ソースコード
今回私はマシンエージェントのソースコード作成にエージェント作成フォーマットの一部を拝借させていただきました。また、別の作り方として、去年卒業された阿部さんやほかの方の論文にも基本部分のソースコードが載っているので、まだ作成をしていない方はそちらの方を使って、自分なりに数値を変えて使用するのも良いかと思われます。
以下がソースコードです

```
package strategy;

import java.util.*;


public class TestStrategy extends Strategy{

        private Random random;
        private int Ave1=0;
        private int Ave2=0;
        private int Ave3=0;
        private int counter=10;
        private int maxPosition = 300;
```

```
public TestStrategy(int seed) {
                random = new Random(seed);
        }


        public Order getOrder(int[] spotPrices, int[] futurePrices, int pos,long
 money, int restDay) {

                                Order order = new Order();
                                setOrderBuySell(order, futurePrices, spotPrices,
pos);
                                setOrderQuant(order, spotPrices, futurePrices, pos);
                                setOrderPrice(order, futurePrices, spotPrices);
                                setMaxPosition(order, futurePrices, pos) ;
                                setMessage(order);
        return order;
        }


        public void setOrderBuySell(Order order, int[] futurePrices,int[]spotPrices, int
pos) {
         Ave1=(spotPrices[119]+spotPrices[118])/2 ;

         int tmp =0;
         for(int i=0; i<counter; i++){

                tmp+=spotPrices[110+i];
```

```
        }
        Ave2=tmp/counter;

                Ave3=(futurePrices[59]+futurePrices[58]+futurePrices[57])/3 ;

                        if (Ave1 > Ave2&&(Ave1-spotPrices[119])<0)     {
                                order.buysell =1 ;
                        } else if ( Ave3 > Ave1 && Ave3 > Ave2){
                                order.buysell =1 ;

                        } else if (Ave1 < Ave2&&(Ave1-spotPrices[119])>0){
                                order.buysell =2 ;
                        } else if ( Ave3 < Ave1 && Ave3 < Ave2){
                                order.buysell =2 ;

                        }else {
                                order.buysell=0 ;
                        }
                }

        public void setOrderQuant(Order order,int[] spotPrices, int[] futurePrices,
int pos) {

                        if (Ave1 > Ave2&&(Ave1-spotPrices[119])<0)     {
                                order.quant = 100 ;
```

```
                } else if ( Ave3 > Ave1 && Ave3 > Ave2){
                        order.quant = 100 ;


                } else if (Ave1 < Ave2&&(Ave1-spotPrices[119])>0){
                        order.quant = 200 ;
                } else if ( Ave3 < Ave1 && Ave3 < Ave2){
                        order.quant = 200 ;
                }


        }



        public void setOrderPrice(Order order, int[] futurePrices, int[]
 spotPrices) {
                int sasine =
 (spotPrices[119]+spotPrices[118]+futurePrices[59]+futurePrices[58])/4 ;
                  order.price = sasine ;
}



        public void setMaxPosition(Order order, int[] futurePrices, int pos){
                if( order.buysell == Order.BUY ) {
                    if( pos > maxPosition ) {
                        order.buysell = Order.NONE ;
                    }
                } else if ( order.buysell == Order.SELL ) {
                    if( pos < -maxPosition ) {
                        order.buysell = Order.NONE ;
                    }
                }
```

```
    }

    public Order setMessage(Order order){

    message(
                        Order.buySellToString(order.buysell)+
                        " price ="+ order.price +
                        "volume =" + order.quant +
                        "(Ave1 =" +Ave1 +
                        ")"
                        );
    return order;

    }
```

}

これも同様に数値、文字を変えるだけで自分なりのマシンエージェントを作ることが出来ます。まだ java に関してやったことが無いし、自分は初心者であるという人は最初は誰かのをコピーして使い、ソースのそれぞれの意味を自分で調べて使うのが良いかと思われます。

3-3　ポイント

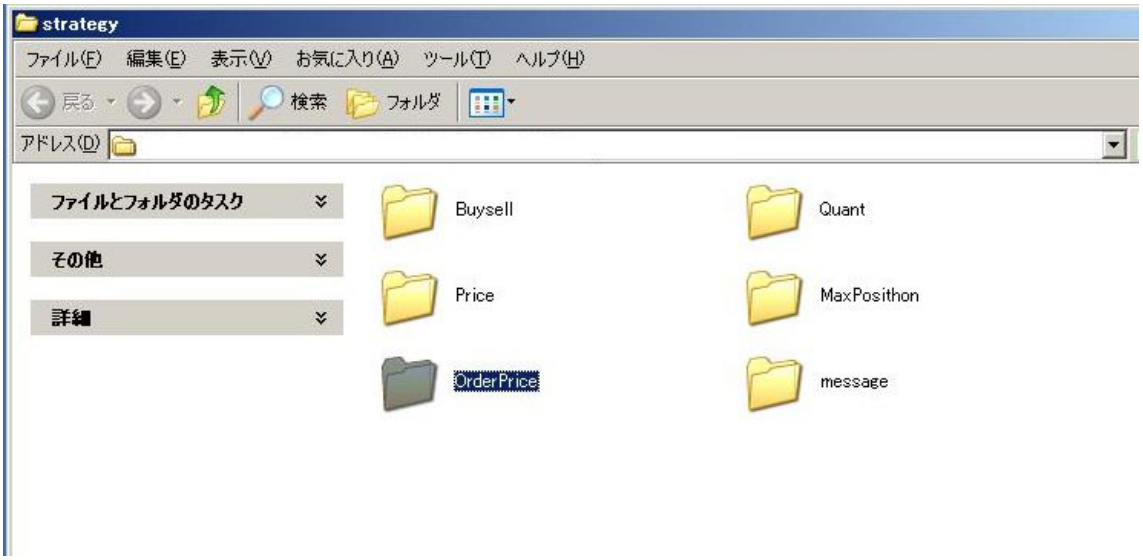
一応、この論文は初心者が読んでいることを前提に進めているので、ソースの各部分のポイント部分を細かく説明して行きたい。基本的に各自決めることはソースの getOrder の中にある様に

setOrderBuySell(order, futurePrices, spotPrices, pos);　売買判断
setOrderQuant(order, spotPrices, futurePrices, pos); その時の数量
setOrderPrice(order, futurePrices, spotPrices);　　その時の価格

この 3 つである。結局マシンエージェントであろうと、ヒューマンエージェントであろうと中心にあるものは注文(Order)する事なので、これが基本となる。

setMaxPosition(order, futurePrices, pos) ;　自分が持つ量の限界

setMessage(order);
MaxPosition は別の所で定義します。Message の部分は後で説明いたします。
まず、形から知ってもらいたい。このプログラムは大きな入れ物に小さないくつかの入れ物が入っていると思っていただければそれでよい。その様子を例として、フォルダで説明すると



このようになる。Stratgy という大きなファイル(これをｃｌａｓｓファイルと言う)の中に細かい動作をしてくれるプログラム(Ｂｕｙｓｅｌｌ やＱｕａｎｔ等)が入って１個のプログラムを形成しているということだ。と思ってくれればよい。

では、内容に入って行きますが、まず一番上の package strategy;の部分は「このプログラムは strategy というファイルに入っています」という事を意味しているので、もしこれが strategy という名前のファイルに入っていない場合はエラーが発生するので注意してもらいたい。

public class TestStrategy extends Strategy　これ以下の private～の部分は「このプログラムの条件や使う文字が入っている」という意味と思っていただければそれで良いと思う。実際ここで、これ以降に使う Ave という文字や、商品の所持限界量などが定義されている。ここで Ave などの文字を宣言しておけば、これ以降のプログラムの中身を定義したときにその定義した場所より下の部分ではまた定義しない限りずっとその内容が適用されるのでここで宣言しておいた方がプログラムが簡素になり、読みやすいプログラムが出来上がる。どういう事かというと、プログラムの set orderbuysell の部分で
Ave1=(spotPrices[119]+spotPrices[118])/2；
と定義している。これが orderbuysell の部分以下ではこの定義が必要なくなるので、Ave1 と入れるだけで(spotPrices[119]+spotPrices[118])/2 と判断してくれる。それゆえ

orderbuysell の部分で定義したため、orderquant の部分でまた Ave を使用するのだが、そこでは、Ave の中身が orderbuysell のと同じで良いのなら定義する事無くいきなり Ave を使ってもエラーが出ない。
ちなみに、ここで counter=10 とあるが、これはこれ以降 counter という文字が出たら 10 と言う数字に変換するという意味を表している。この counter を使用するのはプログラムの以下の部分である。

```
int tmp =0;
for(int i=0; i<counter; i++){

tmp+=spotPrices[110+i];

}
Ave2=tmp/counter;
```

ここでは for 文というのを使った。これは平たい話「繰り返す」という意味を表している。ここでは、まず tmp という文字が使用できるよう定義しておく。1 行目の for～の部分は「 i という数字がありこれは、初期値 0 であり、 1 ずつ増えていき（i++）最終的に 10 以下でその増加が止まるという意味である(i<counter)。そして、その次の行で「tmp は spotPrices[110]から spotPrices[119]までを足した値であるという意味を持つ。tmp の前に+があるのは足す為にある。減算したい場合はここに-を入れればよい。そして、Ave2 はその tmp という数値を 10 で割ったものであるということを最後の部分で定義した。つまり、Ave2 は現物価格の 10 日分の平均値を出したものであると言うわけである。つまり、上の for 文のプログラムと

```
Ave2=(spotPrices[119]+spotPrices[118]+spotPrices+[117]+spotPrices[116]+spotPrices[115]+spotPrices[114]+spotPrices[113]+spotPrices[112]+spotPrices[111]+spotPrices[110])/10;
```

は全く同じ働きをするということである。
ちなみに、Ave1 は最近 2 日の現物価格の平均値を指していて、Ave3 は最近 3 日間の先物価格の平均値である。Ave 1 と 3 にも for 文を使うのはもちろん可能ではあるが、対象の数字が少ない事からそのまま書く事にした。こうした方が for 文の意味を取りやすいと思ったのもそのまま書いた理由の一つである。
そして

```
if (Ave1 > Ave2&&(Ave1-spotPrices[119])<0)    {
```

```
                    order.buysell =1；
                } else if ( Ave3 > Ave1 && Ave3 > Ave2){
                    order.buysell =1；
```

この部分で各設定をしている。上記の部分では、英語が読めれば大体察することが出来るだろうが「もし Ave1 が 2 より大きく、かつ（&&はそういう意味である）Ave1 が直近の現物価格より高い場合は買い注文(1)をするという意味を表している。ちなみに order.buysell の部分に 2 を入れると売る、を意味して 0 の場合は何もしないと言う意味である。
それ以外の場合として、3 行目以降の部分がある。そして、それ以外の事情が発生した場合を考慮して最後に

```
                }else {
                    order.buysell=0；
                }
            }
```

この部分を作り、何もしないとしておく。

そして、orderquant の部分で買う場合の数量を決定している。ここの部分にはもちろん else～は必要ない。例外が起きた場合は何もしないからだ。ここの部分は orderbuysell の部分からコピー&ペーストして作った方が良い。Orderbuysell の部分と違う条件が入っていた場合は売買をしてくれないからである。

そして、最後の部分を見てもらいたい。最後の部分に

```
    message(
                    Order.buySellToString(order.buysell)+
                    " price ="+ order.price +
                    "volume =" + order.quant +
                    "(Ave1 =" +Ave1 +
                    ")"
                    );
    return order;

    }
```

というのがある。これは、マシンエージェントを起動した際に結果を表示させるためにある。これをソース内に入れておくと、どのときに取引を行い、順位はいくつなのか等が出てくる。より良いマシンエージェントを作成するためには必要と思われるので、自分で作

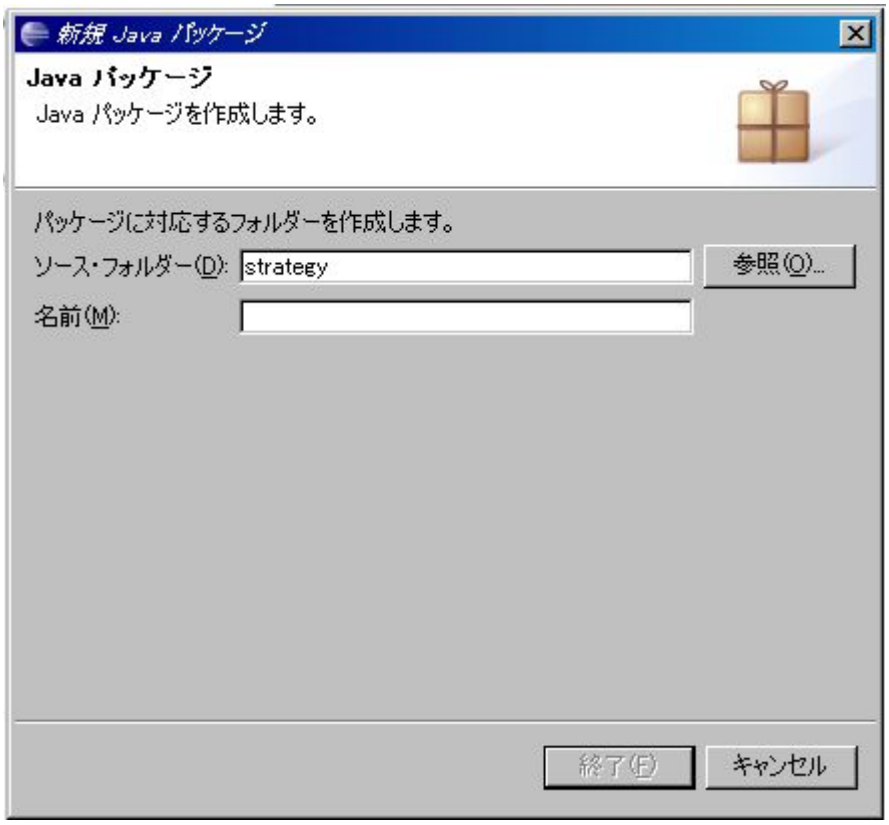
る際は是非入れてもらいたい。

3-4　コンパイル

さて、出来上がったマシンエージェントをコンパイルして、class ファイルを作成しなければ U-mart に入れることは出来ないので、そのやり方をここで説明したいと思います。まずは UMIE フォルダを開き src 内にある strategy フォルダを開いてもらいたい。この中に自分のマシンエージェントを入れる。次にダウンロードした eclipse を起動してもらいたい。起動したら、まず、ファイルをクリックし、新規にカーソルを合わせ、プロジェクトを選択する。そうすると新規プロジェクトウィザードが出てくるので、その中の Java プロジェクトを選択する。すると

この様な画像が出てくる。プロジェクト名は任意で良いので好きなのを入れてもらいたい。（私は strategy としました）
こうする事により、eclipse 右側のパッケージエクスプローラーにプロジェクト名のオブジェが出てくる。

次にこの strategy を右クリックして、新規にカーソルを合わせ、パッケージを選択してもらいたい。すると、以下のウインドウが出てきます。



ここで注意して欲しいのは、この名前の部分を src 内のフォルダ名と同じにしてもらいたい。

つまり、server、guiserver、event、strategy の４つのパッケージを作成して欲しい。そして、それぞれをコピー&ペーストして各パッケージに貼り付ける。そうしないと、エラーが発生して、コンパイルができなくなる。
ちなみに、なぜ eclipse でコンパイルするかと言うと、コマンドプロンプトから javac でコンパイルする事が難しいからである。マシンエージェントはサーバーとリンクしている事を前提として動いているので、コマンドプロンプトからではたとえ正しく書かれたプログラムであろうともエラーが発生してしまうからである。
貼り付けたら、eclipse は勝手にコンパイルしてくれるが、明示的にコンパイルしたい場合は、上のプロジェクトを選択して、すべてビルドをクリックすればコンパイルしてくれる。
以上で、作成は完了である。

4：最後に

今回、卒論を書くにあたってマシンエージェントを作成したが、自分としては至らなかった所が多数ある論文となってしまった。それをこの場を借りて、私の論文を読んでいただいた方々に謝罪したい。

最初は動くマシンエージェントだったら何でもよいと思い論文を作成していたので、その目標は達成する事がなんとか出来たものの、検証などの深いところまでは触れる事が出来なかった。もし、私の論文でマシンエージェントに興味をもってくれたのであるのなら、諸先輩方の論文を是非見ていただきたい。そして、その論文を利用して、もっとも利益の出るマシンエージェントを、皆さんには作っていただきたい。そして、この論文を作成するに当たり、さまざまな事を教えてくれた篠原君には本当に感謝しています。ありがとうございました。

今後の有賀ゼミの発展を祈りつつ、この論文の締めとさせていただきます。みなさん、ここまで読んでいただき本当にありがとうございました。

参考文献（含ホームページ）

超図解 java ルールブック　電通国際情報サービス著　エクスメディア編

http://www.geocities.jp/turtle_wide/index.html

http://javafaq.jp/index.html