MetaQuotes Language 4
Articole

In aceasta sectiune puteti gasi articole care trateaza diferite aspecte ale crearii sau utilizatii expert advisor-ilor sau indicatorilor proprii. Inainte de a incepe sa va creati propriul program MQL4, cititi cu atentie aceste articole. Veti gasi raspunsuri la intrebarile dumneavoastra aici.

- **Caracteristici ale Expert Advisor-i**
- **Caracteristici ale indicatorilor personalizati**
- **Testare de strategie: Modalitati de modelare in timpul testarii**
- **Testare de caracteristici si limite in MetaTrader 4**

Impreuna cu expert advisori si indicatori personalizati, este posibil de asemenea sa creati si sa utilizati scripturi si biblioteci. Principala diferenta intre un script si un expert advisor este modul in care sunt utilizati: Expert Advisor-ii sunt utilizati pentru fiecare modificare a pretului, in timp ce Scripturile sunt executate doar o data la cerere. De exemplu, cu un script puteti implementa secventa variata pentru terminalul client, afisare de mesaje suplimentare sau asteptare pentru actiuni din partea utilizatorului.

Bibliotecile sunt utilizate pentru a crea arhive cu functii proprii. Aceasta inseamna ca un dezvoltator de soft poate colectiona functiile originale ale sale in fisiere din biblioteca si sa le apeleze pentru orice program personalizat. Din momentul in care o biblioteca este creata, toate functiile descrise in aceasta sunt compilate si salvate pentru o utilizare viitoare. Trebuie mentionat ca functiile care nu sunt apelate sunt scoase din codul expert advisor-ilor, indicatorilor personalizati si scripturilor in timpul procesului de compilare pentru a-l optimiza. Aceasta este particularitatea crearii de biblioteci in MQL4 fata de alte programe.

Caracteristici ale crearii Expert Advisori

Crearea unui expert advisor in sistemul de tranzactionare MetaTrader are o serie de caracteristici.

- Inainte de dechiderea unei pozitii trebuie sa verificati daca sunt bani disponibili in cont. Daca nu sunt bani suficienti in cont, operatiunea de deschidere a unei pozitii nu se va indeplini cu succes. Valoarea „FreeMargin" trebuie sa fie mai mica de 1000 doar in timpul testelor deoarece in timpul testelor pretul unui lot este 1000.

```
if(AccountFreeMargin() < 1000) return(0);   // not enough money
```

- Puteti accesa datele istorice folosing vectorii predefiniti Time, Open, Low, High, Close, Volume . Din motive istorice, indicii acestor vectori cresc de la capat la inceput. Inseamna ca cele mai noi date au index 0. Index 1 indica datele mutate cu o perioada inapoi, index 2 inseamna datele mutate cu doua perioade inapoi, 3 este cu trei perioade inapoi samd.

```
// if Close pe bara anterioara este mai mic decat
// Close decat bara anterioara
```

```
if(Close[1] < Close[2]) return(0);
```

- Este posibil de asemenea sa accesezi datele istorice folosind ale intervale de timp si chiar folosing si alte perechi valutare. Pentru a obtine astfel de date, este necesar sa defineste un vector cu o singura dimensiune si sa efectuati o operatiune de copiere folosing functia "ArrayCopySeries". Luati in considerare faptul ca prin apelarea functiei este posibil sa treaca un numar mic de parametri si sa nu se specifice parametrul default.

```
double eur_close_m1[];
int number_copied = ArrayCopySeries(eur_close_m1, MODE_CLOSE, "EURUSD",
PERIOD_M1);
```

In procesul scrierii unui expert advisor, precum si in cazul oricarui alt software, este cateodata necesar sa obtii informatii suplimentare de depanare. Limbajul MQL4 furnizeaza cateva metode de obtinere a unor astfel de informatii.

- Functia „Alert" afiseaza o casuta de dialog cu date definite de utilizator.

```
Alert("FreeMargin grows to ", AccountFreeMargin(), "!");
```

- Functia „Comment" afiseaza date definite de utilizator in coltul stanga-sus al graficului. Secventa de caractere "\n" este utilizata pentru a incepe un nou rand.

```
Comment("FreeMargin is ", AccountFreeMargin(), ".");
```

- Functia „Print" salveaza date definite de utilizator in jurnalul sistemului.

```
Print("FreeMargin is ", AccountFreeMargin(), ".");
```

- Pentru a obtine informatii despre erorile din program, functia „GetLastError" este foarte utila. De exemplu, o operatie cu o ordine intotdeauna returneaza numarul de identificare. Daca numarul de identificare este egal cu 0 (s-a produs o eroare in procesul executarii operatiei), este necesara apelarea functiei „GetLastError" pentru a obtine mai multe informatii despre eroare:

```
int iTickNum = 0;
int iLastError = 0;
...
iTickNum = OrderSet (OP_BUY, g_Lots, Ask, 3, 0, Ask + g_TakeProfit *
g_Points, Red);
if (iTickNum <= 0)
  {
   iLastError = GetLastError();
   if (iLastError != ERROR_SUCCESS) Alert("Some Message");
  }
```

Trebuie sa tineti cont de faptul ca apelarea functiei „GetLastError" afiseaza codul ultimei erori si ii reseteaza valoarea. Din acest motiv, apelarea acestei functii din nou imediat va returna intotdeauna valoarea 0.

- Cum se defineste inceputului unei noi bare? (Poate fi necesar sa aflati ca bara anterioara tocmai a fost terminata.) Exista cateva metode sa faceti acest lucru.

Prima metoda este verificarea numarului de bare:

```
int prevbars = 0;
```

```
...
if(prevbars == Bars) return(0);
prevbars = Bars;
...
```

Aceasta metoda poate sa nu functioneze in timp ce incarcati istoricul. Adica, numarul de bare este schimbat in timp ce „anterioara" nu a fost terminata inca. In acest caz puteti face verificarea mai complicata prin introducerea unei verificari de diferenta intre valori egale cu una.

Urmatoarea metoda se bazeaza pe faptul ca valoarea „Volume" este generata in functie de numarul de miscari care sunt pentru fiecare bara, iar prima miscare inseamna ca valoarea „Volume" pentru o noua bara este egala cu 1:

```
if( Volume > 1) return(0);
...
```

Aceasta metoda poate sa nu functioneze atunci cand sunt multe miscari de pret. Problema este ca miscarile de pret care intra sunt procesate intr-un spatiu separat. Iar daca acest spatiu este ocupat atunci cand vine o alta modificare de pret, aceasta noua miscare de pert nu este procesata pentru a evita supraincarcarea procesorului! In acest caz puteti sa faceti verificarea si mai complicata prin salvarea valorii „Volume".

A treia metoda se bazeaza pe durata cat o bara este deschisa:

```
datetime prevtime=0;
...
if(prevtime == Time[0]) return(0);
prevtime = Time[0];
...
```

Este metoda cea mai de incredere. Functioneaza in toate cazurile.

- Exemplu de lucru cu un fisier de tipul "CSV":

```
int h1;
h1 = FileOpen("my_data.csv", MODE_CSV | MODE_WRITE, ";");
if(h1<0)
  {
   Print("Unable to open file my_data.csv");
   return(false);
  }
FileWrite(h1, High[1], Low[1], Close[1], Volume[1]);
FileClose(h1);
```

- Cateva explicatii referitoare la cod. Fisierul de format „CSV" este deschis primul. In cazul in care apare o problema la deschiderea fisierului, programul se inchide. In cazul in care fisierul se deschide cu succes, se goleste continutul sau, se salveaza date in el, iar fisierul se inchide. Daca doriti sa tineti deschis continutul fisierului, trebuie sa folositi modul de deschidere MODE_READ:

```
int h1;
h1 = FileOpen("my_data.csv", MODE_CSV | MODE_WRITE | MODE_READ, ";");
if(h1<0)
 {
  Print("Unable to open file my_data.csv");
  return(false);
```

```
  }
 FileSeek(h1, 0, SEEK_END);
 FileWrite(h1, High[1], Low[1], Close[1], Volume[1]);
 FileClose(h1);
```

- In acest exeplu, datele sunt adaugate la sfarsitul fisierului. Pentru a efectua acest lucru, folosim functia „FileSeek” imediat dupa ce a fost deschis.

Caracteristici ale crearii Indicatorilor Personalizati

Crearea indicatorilor personalizati in MetaTrader are o serie de caracteristici.

- Pentru ca un program sa fie considerat un indicator personalizat, trebuie sa se incadreze intr-una dintre definitiile:

```
#property  indicator_chart_window      // un indicator este elaborat in principala fereastra a graficului
```

sau

```
#property  indicator_separate_window   // un indicator este elaborat intr-o fereastra separata
```

- Pentru a stabili scala ferestrei pentru un indicator separat, se foloseste urmatoarea definitie:

```
#property  indicator_minimum Min_Value
#property  indicator_maximum Max_Value
```

unde "Min_Value" si "Max_Value" sunt valorile corespunzatoare. De exemplu, aceste valori trebuie sa fie 0 si, respectiv, 100 pentru indicatorul personalizat RSI.

- Numarul inficatorului sir de caractere pentru desenarea unui indicator trebuie definit dupa cum urmeaza:

```
#property  indicator_buffers N
```

unde N poate lua valori de la 1 la 8.

- Culoarea liniilor intr-un indicator poate fi stabilita prin urmatoarele definitii:

```
#property  indicator_color1  Silver
#property  indicator_color2  Red
...
#property  indicator_colorN  <SomeColor>
```

unde N este numarul indicatorului sir de caractere definit prin "#define indicator_buffer".

- Acestea sunt functii care permit controlul procesului calcularii si vizualizatii indicatorului. Indicatorul personalizat al lui Ishimoku Kinko Hyo este folosit aici pentru ilustrare:

```
//+------------------------------------------------------------------+
//|                                                  Ichimoku.mq4 |
```

```
//|                  Copyright © 2004, MetaQuotes Software Corp. |
//|                                       http://www.metaquotes.net/ |
//+------------------------------------------------------------------+
#property copyright "Copyright © 2004, MetaQuotes Software Corp."
#property link      http://www.metaquotes.net/

#property indicator_chart_window
#property indicator_buffers 7
#property indicator_color1 Red
#property indicator_color2 Blue
#property indicator_color3 SandyBrown
#property indicator_color4 Thistle
#property indicator_color5 Lime
#property indicator_color6 SandyBrown
#property indicator_color7 Thistle
//---- input parameters
extern int Tenkan=9;
extern int Kijun=26;
extern int Senkou=52;
//---- indicator buffers
double Tenkan_Buffer[];
double Kijun_Buffer[];
double SpanA_Buffer[];
double SpanB_Buffer[];
double Chinkou_Buffer[];
double SpanA2_Buffer[];
double SpanB2_Buffer[];
//---- span_a drawing begin
int a_begin;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int init()
  {
//----
   SetIndexStyle(0,DRAW_LINE);
   SetIndexBuffer(0,Tenkan_Buffer);
   SetIndexDrawBegin(0,Tenkan-1);
   SetIndexLabel(0,"Tenkan Sen");
//----
   SetIndexStyle(1,DRAW_LINE);
   SetIndexBuffer(1,Kijun_Buffer);
   SetIndexDrawBegin(1,Kijun-1);
   SetIndexLabel(1,"Kijun Sen");
//----
   a_begin=Kijun; if(a_begin<Tenkan) a_begin=Tenkan;
   SetIndexStyle(2,DRAW_HISTOGRAM,STYLE_DOT);
   SetIndexBuffer(2,SpanA_Buffer);
   SetIndexDrawBegin(2,Kijun+a_begin-1);
   SetIndexShift(2,Kijun);
   SetIndexLabel(2,NULL);
   SetIndexStyle(5,DRAW_LINE,STYLE_DOT);
```

```
   SetIndexBuffer(5,SpanA2_Buffer);
   SetIndexDrawBegin(5,Kijun+a_begin-1);
   SetIndexShift(5,Kijun);
   SetIndexLabel(5,"Senkou Span A");
  //----
   SetIndexStyle(3,DRAW_HISTOGRAM,STYLE_DOT);
   SetIndexBuffer(3,SpanB_Buffer);
   SetIndexDrawBegin(3,Kijun+Senkou-1);
   SetIndexShift(3,Kijun);
   SetIndexLabel(3,NULL);
   SetIndexStyle(6,DRAW_LINE,STYLE_DOT);
   SetIndexBuffer(6,SpanB2_Buffer);
   SetIndexDrawBegin(6,Kijun+Senkou-1);
   SetIndexShift(6,Kijun);
   SetIndexLabel(6,"Senkou Span B");
  //----
   SetIndexStyle(4,DRAW_LINE);
   SetIndexBuffer(4,Chinkou_Buffer);
   SetIndexShift(4,-Kijun);
   SetIndexLabel(4,"Chinkou Span");
  //----
   return(0);
  }
  //+------------------------------------------------------------------+
  //| Ichimoku Kinko Hyo                                               |
  //+------------------------------------------------------------------+
  int start()
    {
     int    i,k;
     int    counted_bars=IndicatorCounted();
     double high,low,price;
  //----
     if(Bars<=Tenkan || Bars<=Kijun || Bars<=Senkou) return(0);
  //---- initial zero
     if(counted_bars<1)
       {
        for(i=1;i<=Tenkan;i++)    Tenkan_Buffer[Bars-i]=0;
        for(i=1;i<=Kijun;i++)     Kijun_Buffer[Bars-i]=0;
        for(i=1;i<=a_begin;i++) { SpanA_Buffer[Bars-i]=0; SpanA2_Buffer
[Bars-i]=0; }
        for(i=1;i<=Senkou;i++)  { SpanB_Buffer[Bars-i]=0; SpanB2_Buffer
[Bars-i]=0; }
       }
  //---- Tenkan Sen
     i=Bars-Tenkan;
     if(counted_bars>Tenkan) i=Bars-counted_bars-1;
     while(i>=0)
       {
        high=High[i]; low=Low[i]; k=i-1+Tenkan;
        while(k>=i)
          {
           price=High[k];
```

```
                  if(high<price) high=price;
                  price=Low[k];
                  if(low>price)  low=price;
                  k--;
                  }
                Tenkan_Buffer[i]=(high+low)/2;
                i--;
                }
              //---- Kijun Sen
                i=Bars-Kijun;
                if(counted_bars>Kijun) i=Bars-counted_bars-1;
                while(i>=0)
                  {
                  high=High[i]; low=Low[i]; k=i-1+Kijun;
                  while(k>=i)
                    {
                    price=High[k];
                    if(highprice)  low=price;
                    k--;
                    }
                  Kijun_Buffer[i]=(high+low)/2;
                  i--;
                  }
              //---- Senkou Span A
                i=Bars-a_begin+1;
                if(counted_bars>a_begin-1) i=Bars-counted_bars-1;
                while(i>=0)
                  {
                  price=(Kijun_Buffer[i]+Tenkan_Buffer[i])/2;
                  SpanA_Buffer[i]=price;
                  SpanA2_Buffer[i]=price;
                  i--;
                  }
              //---- Senkou Span B
                i=Bars-Senkou;
                if(counted_bars>Senkou) i=Bars-counted_bars-1;
                while(i>=0)
                  {
                  high=High[i]; low=Low[i]; k=i-1+Senkou;
                  while(k>=i)
                    {
                    price=High[k];
                    if(high<price) high=price;
                    price=Low[k];
                    if(low>price)  low=price;
                    k--;
                    }
                  price=(high+low)/2;
                  SpanB_Buffer[i]=price;
                  SpanB2_Buffer[i]=price;
                  i--;
                  }
```

```
//---- Chinkou Span
  i=Bars-1;
  if(counted_bars>1) i=Bars-counted_bars-1;
  while(i>=0) { Chinkou_Buffer[i]=Close[i]; i--; }
//----
  return(0);
 }
//+------------------------------------------------------------------+
```

- Functia "SetIndexStyle" controleaza desenarea parametrilor unui indicator sir de caractere. Modul de desen DRAW_LINE presupune ca sunt desenate liniile dintre valorile definite intr-un indicator sir de caractere corespunzator. Modul de desen DRAW_HISTOGRAM fiind aplicat in indicatorul din fereastra principala are caracteristicile sale speciale, se asemenea. O histograma este desenata intre valorile corespunzatoare ale indicilor celor doua siruri de caractere: una para (aici: SpanA_Buffer) si una impara (aici: SpanB_Buffer). Se foloseste valoarea cea mai mare pentru culoarea indicelul sir de caractere.
- Functia "SetIndexDrawBegin" specifica unde incepe elementul datelor semnificative al indicatorul sir de caractere.
- Functia "SetIndexBuffer" permite declararea oricarui sir de caractere cu o dimensiune al unui tip „double” drept un index sir de caractere. Asadar, sistemul va administra indici sir de caractere. Acesta este motivul pentru care marimea acestor siruri de caractere nu trebuie specificata.

```
//---- indicator buffers
double Tenkan_Buffer[];
double Kijun_Buffer[];
double SpanA_Buffer[];
double SpanB_Buffer[];
double Chinkou_Buffer[];
double SpanA2_Buffer[];
double SpanB2_Buffer[];
```

- Functia ArraySize nu poate fi aplicata indicatorului sir de caractere, deoarece este inutila. Este de asemenea inutil sa aplici functia ArrayInitializa indicatorilor sir de caractere, in special ca functie ‚init', atunci cand indicatorii sir de caractere nu au fost inca alocati. Indicatorii sir de caractere sunt initializati automat in timpul alocarii si realocarii memoriei. Sunt folisite pentru initializarea valorilor EMPTY_VALUE sau valoarea specificata de functia SetIndexEmptyValue. Valorile „Empty” nu sunt afisate.
- Functia „SetIndexLabel” seteaza numele care va fi afisat in tool tips si in fereastra cu date impreuna cu valoarea corespunzatoare („ValueE” este stabilita default, unde N este numarul indexului sirului de caractere). Daca NULL este transferat in loc de nume, valoarea corespunzatoare nu va fi afisata nici in tool tips, nici in fereastra de date. In acest caz, norii de puncte sunt hasurati folosind o histograma si sunt limitati la o linie. La aceasta, valorile corespunzatoare ale „line” si „histogram” ale sirului de numere sunt aceleasi si este posibila afisarea doar a unuia dintre ele.
- Functia „IndicatorCounter” permite organizarea calculului economic al unui indicator. Functia returneaza numarul barelor pana la momentul de dinaintea lansarii indicatorului, adica, numarul barelor care sunt deja calculate (potentiale, daca nu au existat erori sau incheiere prematura in timpul lansarii precedente) care nu au nevoie de nici o recalculare. La reinitializarea indicatorului personalizat sau actualizarea semnificativa a datelor istorice, aceasta valoare va fi in mod automat resetata la 0.

- Sa discutam inca un exemplu. Indicatorul persoanalizat denumit Accelerator/Decelerator Oscillator:

```
//+------------------------------------------------------------------+
//|                                                Accelerator.mq4 |
//|                  Copyright © 2005, MetaQuotes Software Corp. |
//|                                       http://www.metaquotes.net/ |
//+------------------------------------------------------------------+
#property  copyright "Copyright © 2005, MetaQuotes Software Corp."
#property  link      http://www.metaquotes.net/
//---- indicator settings
#property  indicator_separate_window
#property  indicator_buffers 3
#property  indicator_color1  Black
#property  indicator_color2  Green
#property  indicator_color3  Red
//---- indicator buffers
double     ExtBuffer0[];
double     ExtBuffer1[];
double     ExtBuffer2[];
double     ExtBuffer3[];
double     ExtBuffer4[];
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
int init()
  {
//---- 2 additional buffers are used for counting.
   IndicatorBuffers(5);
//---- drawing settings
   SetIndexStyle(0,DRAW_NONE);
   SetIndexStyle(1,DRAW_HISTOGRAM);
   SetIndexStyle(2,DRAW_HISTOGRAM);
   IndicatorDigits(Digits+2);
   SetIndexDrawBegin(0,38);
   SetIndexDrawBegin(1,38);
   SetIndexDrawBegin(2,38);
//---- 4 indicator buffers mapping
   SetIndexBuffer(0,ExtBuffer0);
   SetIndexBuffer(1,ExtBuffer1);
   SetIndexBuffer(2,ExtBuffer2);
   SetIndexBuffer(3,ExtBuffer3);
   SetIndexBuffer(4,ExtBuffer4);
//---- name for DataWindow and indicator subwindow label
   IndicatorShortName("AC");
   SetIndexLabel(1,NULL);
   SetIndexLabel(2,NULL);
//---- initialization done
   return(0);
  }
//+------------------------------------------------------------------+
//| Accelerator/Decelerator Oscillator                               |
```

```
//+------------------------------------------------------------------+
int start()
  {
   int    limit;
   int    counted_bars=IndicatorCounted();
   double prev,current;
//---- last counted bar will be recounted
   if(counted_bars>0) counted_bars--;
   limit=Bars-counted_bars;
//---- macd counted in the 1-st additional buffer
   for(int i=0; i<limit; i++)
      ExtBuffer3[i]=iMA(NULL,0,5,0,MODE_SMA,PRICE_MEDIAN,i)-
                    iMA(NULL,0,34,0,MODE_SMA,PRICE_MEDIAN,i);
//---- signal line counted in the 2-nd additional buffer
   for(i=0; i=0; i--)
     {
      current=ExtBuffer3[i]-ExtBuffer4[i];
      prev=ExtBuffer3[i+1]-ExtBuffer4[i+1];
      if(current>prev) up=true;
      if(current<prev) up=false;
      if(!up)
        {
         ExtBuffer2[i]=current;
         ExtBuffer1[i]=0.0;
        }
      else
        {
         ExtBuffer1[i]=current;
         ExtBuffer2[i]=0.0;
        }
      ExtBuffer0[i]=current;
     }
//---- done
   return(0);
  }
//+------------------------------------------------------------------+
```

- Functia „IndicatorBuffers” specifica numarul de spatii de diferentiere de utilizat pentru calcularea indicatorilor. In general, aceasta functie este apelata daca sunt utilizati mai multi indici sir de caractere decat este necesar pentru desenarea unui indicator. La aceasta, sistemul administreaza siruri de caractere suplimentare.
- Functia „SetIndexDigits” administreaza precizia output-ului informational. In acest caz, cand diferenta dintre doua medii mobile precum si diferenta viitoare dintre rezultat si linia de semnal este calculata, precizia standard de 4 cifre dupa punct este evident insuficienta.
- Functia „SetIndexDrawBegin” specifica unde incepe elementul datelor semnificate ale indicatorului sir de caractere. In exemplul nostru, linia semnal este calculata ca medie mobila simpla a unei alte medii mobile simple. Din acest motiv, primele 38 de valori ale indicatorului sunt considerate goale si nu se deseneaza.
- Functia „IndicatorShortName” stabileste un asa-numit nume scurt al indicatorului care va fi afisat in coltul stanga sus in fereastra de indicator sin in „DataWindow”. Daca numele scurt nu a fost setat, numele indicatorului personalizat va fi utilizat ca precedentul. In exemplul dat, nu este

necesara utilizarea functiei SetIndexLabel deoarece doar o singura valoare este rezultata. Deci, numele indicatorului este suficient pentru rezultarea unei singure valori.

- Functia „SetIndexStyle" administreaza parametrii de desen al unui indicator sir de caractere. Modul de desen al DRAW_NONE inseamna ca linia nu trebuie desenata. Problema este ca histograma indicatorului prezentat trebuie colorata in 2 culori diferite. Datele din ExtBuffer0 sunt alocate in alte doua siruri de caractere, ExtBuffer1 si ExtBuffer2. Pentru a nu avea ca rezultat date duble in tool tips sau in fereastra de date, se utilizeaza functia SetIndexLabe cu parametrul NULL. Modul de desen al DRAW_HISTROGRAM fiind aplicat unui indicator dintr-o fereastra separata, permite desenarea histogramei intre valorile zero si cea corespunzatoare sirului de caractere (comparati cu desenarea unei histograme in fereastra principala descris anterior).
- Parametrii de input (intrare) utilizati pentru calculul indicatorilor si functiilor personalizate trebuie definiti ca „extern" si pot fi de diferite tipuri.
- Daca parametrilor de input nu le-au fost setati indicatorii personalizati corespunzatori, vor fi apelati in formatul cel mai simplu.

```
double current_AC = iCustom( NULL, 0, "Accelerator", 0, 0 );
```

Transferul celor doua valori „NULL" si „0" inseamna ca graficul prezent va fi folosit. Numele fisierului corespunzator (fara extensia mq4) este folosit drept nume pentru indicator personalizat. Daca ultimul in afara de un parametru este 0, inseamna ca suntem interesati de date de la primul indicator sir de caractere. Ultimul parametru fiind 0, inseamna ca suntem interesati de valoarea ultimului element (adica, valoarea cea mai recenta, prezenta) al indicatorului sir de caractere solicitat.

- Parametrii sunt transferata in functia calculului indicatorului pesonalizat in ordinea in care au fost descrisi. De exemplu, indicatorul personalizat denumit „Ichimoku" si avand parametrii (9,26,52) vor fi apelati dupa cum urmeaza:

```
iCustom( NULL, 0, "Ichimoku", 9, 26, 52, 0, shift );
```

Strict vorbind, parametrii indicator sir de caractere nu trebuie in mod neaparat transferati functiei. Daca nici o variabila externa nu este definita in program, este inutil transferul parametrilor. Sau, daca este necesar, valorile initiale pot fi utilizate in descrierea parametrilor. De exemplu, acelasi indicator sir de caractere fara parametri va fi apelat dupa cum urmeaza:

```
iCustom( NULL, 0, "Ichimoku", 0, shift );
```

Aceasta inseamna ca vor fi utilizate valori care initializeaza variabile "Tenkan", "Kijun", "Senkou", adica 9, 26 si 52. Cu toate acestea, daca un indicator personalizat avand diferite seturi de parametri este apelat intr-un Expert Advisor, este recomandata utilizarea setarilor standard.

Trebuie mentionat faptul ca surplusul de indicatori personalizati precum si cei scrisi incorect vor diminua semnificativ viteza de lucru a terminalului client!

Testare de strategie: Modalitati de modelari in timpul testarii

Introducere

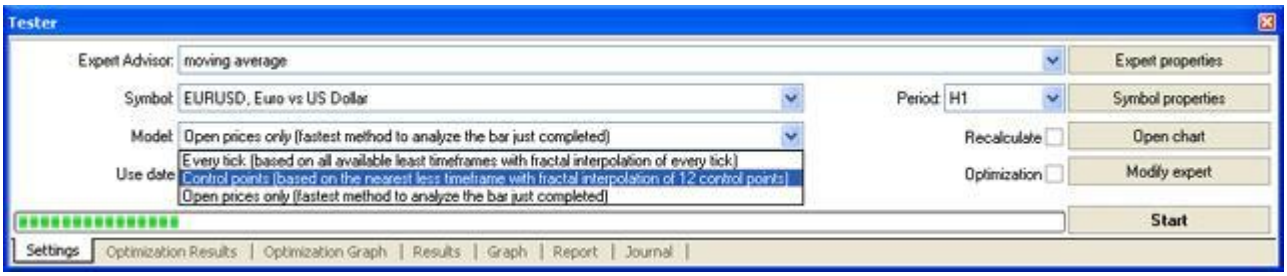
Multe programe de analiza tehnica permit testarea strategiilor de tranzactionare cu date instorice. In cele mai multe cazuri, testele sunt efectuate pe date deja completate fara nici o incercare de a modela trendurile fara o bara a pretului. Era mult mai rapid, dar nu suficient de precis.

Este important de ales un mod relevand de dezvoltare a modelarii barelor de pret penru a realiza o testare de calitate a strategiilor de tranzactionare.

Intr-adevar, nu poate exista o situatie ideala unde sa existe un istoric complet al miscarii preturilor pentru o testare extrem de precisa. Este foarte dificil pentru un trader sa gaseasca istoricul complet al miscarii pretului pentru o perioada de cativa ani, pentru a-l analiza.
Pentru a solutiona aceasta problema, datele unor perioade mai precise pot fi utilizate ca puncte de referinta cu schimbari ale modelarii preturilor intre ele.

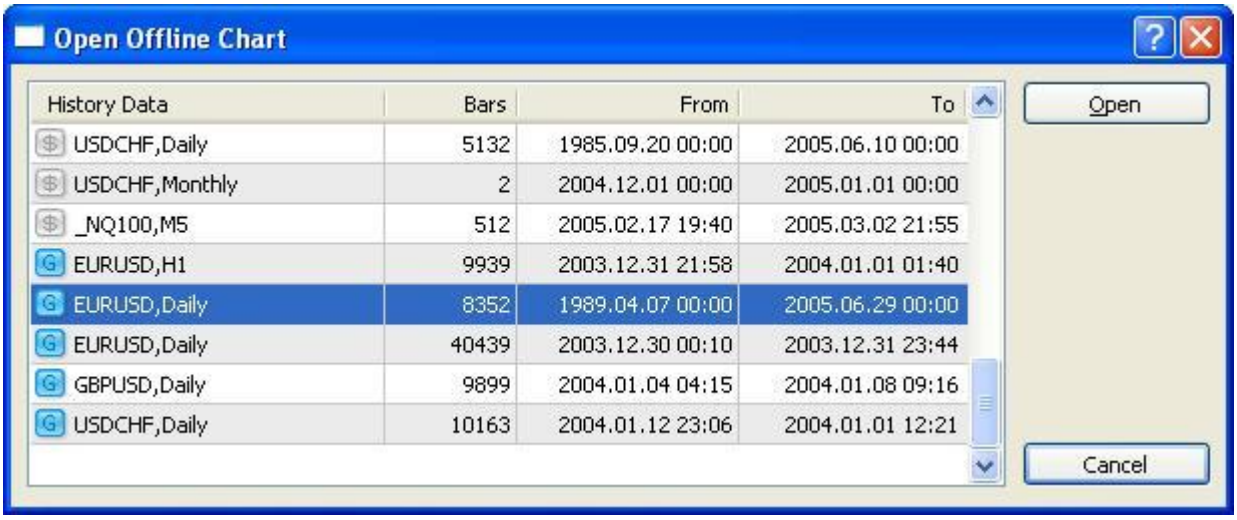
Modalitati de modelare a barelor de pret

Exista trei modalitati de dezvoltare a modelarii barelor utilizate in MetaTrader 4 Client Terminal:



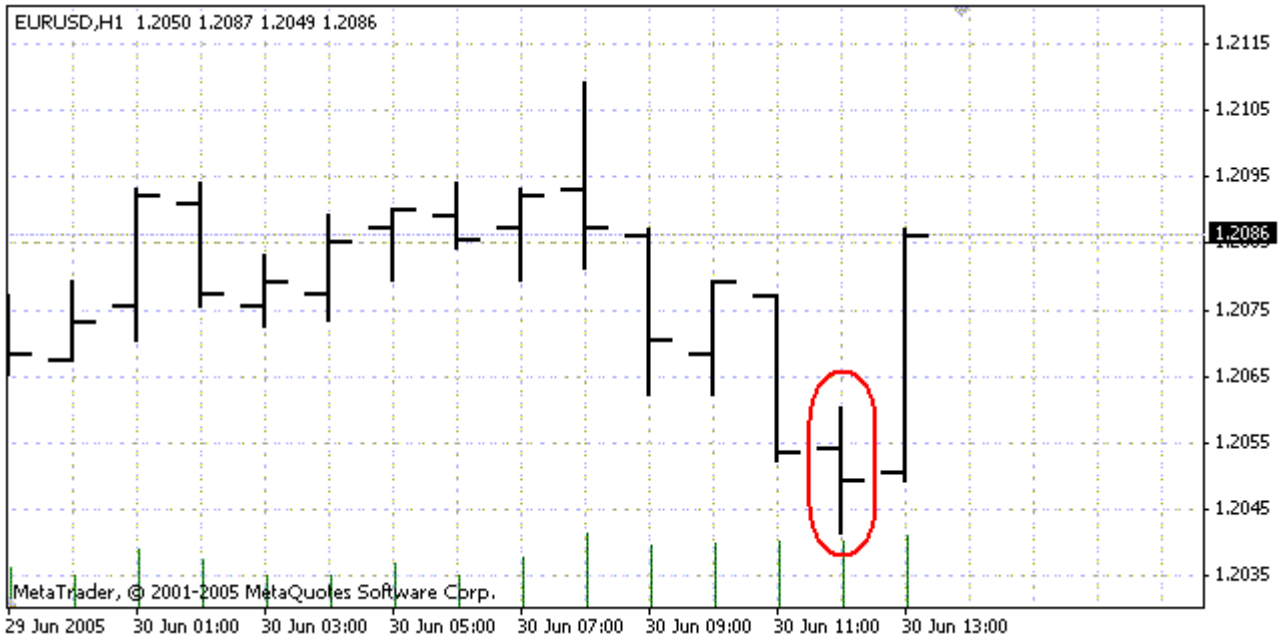
- Every tick – (fiecare miscare de pret) - (bazat pe toate disponibile mai putin timeframes cu interpolare fractala pe fiecare miscare de pret)
- Control points – (puncte de control) - (sunt utilizate cei mai apropiati timeframe cu interpolara fractala)
- Open prices – (preturi deschise) -(modalitate rapida pentru barele completate)

Barele de pret intermediare sunt generate inainte de inceperea testarii, rezultatele fiind salvate in fisier (de exemplu: **/tester/history/eurusd_1440_1.fxt**). Datele cuprinse in acest mod permit dupa aceeas accelararea foarte mare a testerului. Dupa ce a fost activat „Recalculate”, recalcularea datelor intermediare poate fi efectuata.
Utilizarea datelor cuprinse anterior permit formarea testelor pe baza propriilor date intermediare. Pentru a face acest lucru, este suficienta salvarea fisierului in formatul adecvat (format *.FXT, deschis complet) in directorul **/tester/history/**. Aceste fisiere sunt usor de deschis in terminal precum grafice offline prin **File -> Open offline**.



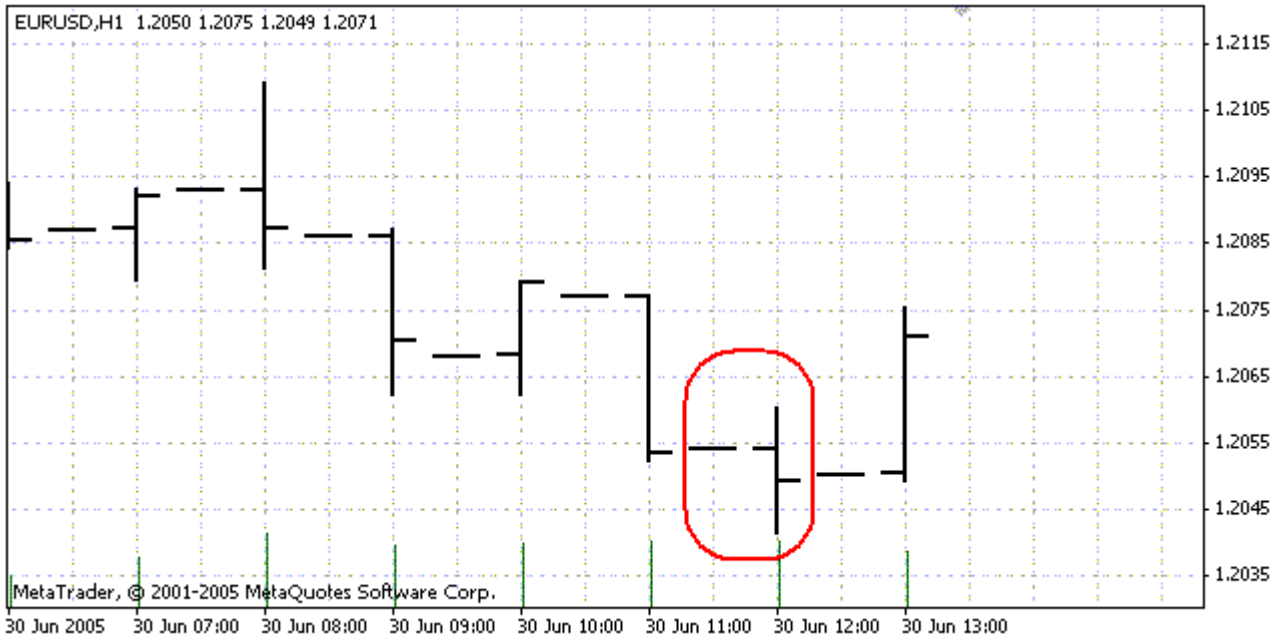
Exemple de modelare

Sa incepem cu cea mai simpla metoda de modelare bazate pe un grafic ora de ora. Acum, sa studiem bara de ora (Iunie, 30 2005, 12:00) incercuita cu rosu:



Open Price (Pret deschis)

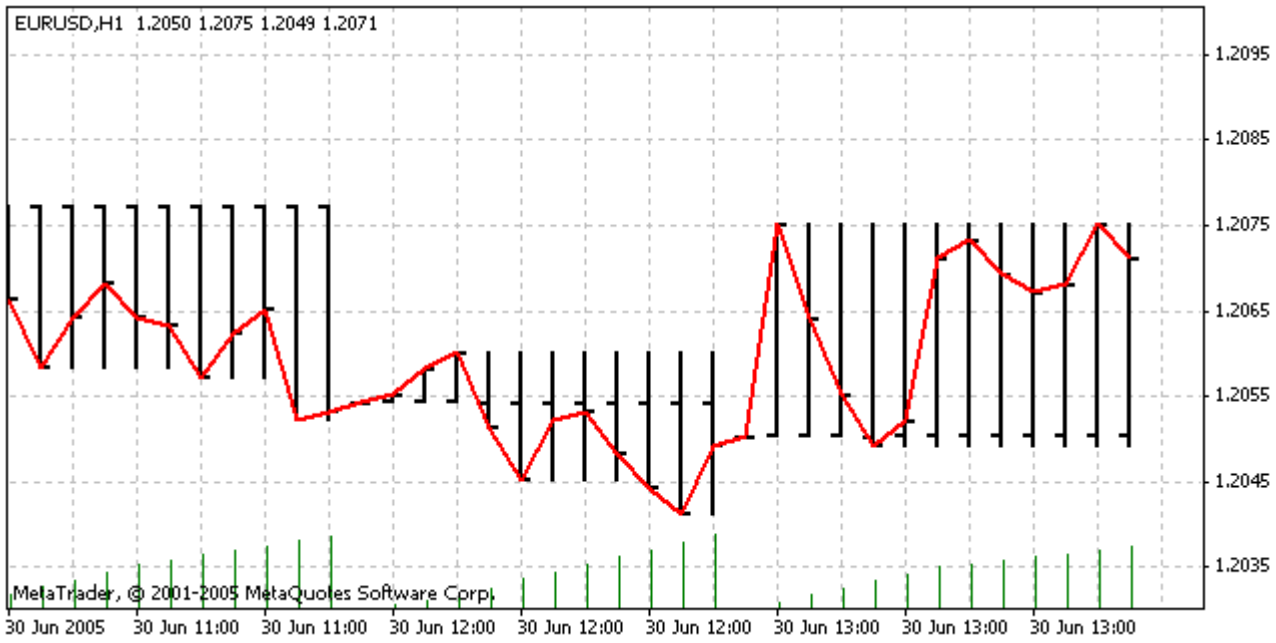
Anumiti traderi nu doresc sa depinda de particularitatile modelarii intra-bara si creaza experts trading pe barele completate. Faptul ca bara curenta de pret este completata in intregime poate fi stiu doar cand apare urmatoarea. Acestia sunt expertii pentru care modul modelarii „Open price” este intentionat.



In acest mod, bara este deschisa (Open = High = Low = Close, Volume=1), acesta permitand expertilor sa identifice sfarsitul completarii barei anterioare de pret. In aceasta bara incipienta este lansata testarea expertilor. In etapa urmatoare, bara curenta, completata in intregime, este generata, dar nu se testeaza nimic pe ea!

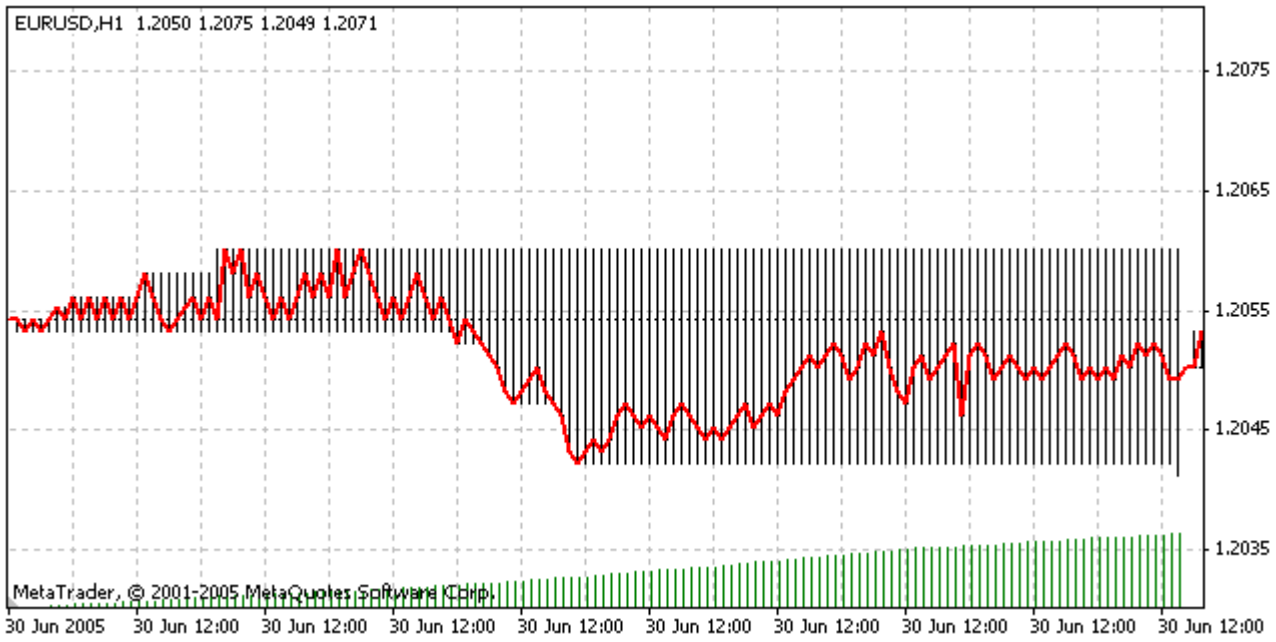
Control Points (puncte de control)(cel mai apropiat less timeframe)

Metoda de modelare a punctelor de control are scopul estimarii brute al experts trading intr-o bara. Pentru a utiliza aceaste metoda, datele istorice ale celui mai apropiat less timeframe trebuie sa fie disponibil. In cele mai multe cazuri, datele unui timeframe mai mic disponibil nu acopera complet intervalul de timp al timeframe testat. Daca nu exista date pentru un timeframe mai mic, dezvoltarea barei este generata pe baza preturilor inchise ale precedentelor 12 bare. Adica, miscarea dintr-o bara repeta miscarea pretului din ultimele 12 timeframe-uri, din acest motiv se numeste interpolare fractala.
Aceasta metoda de generare este total diferita de metoda de transfer de date utilizata in versiunile precedente ale Terminalului Client, din moment ce acea metoda permitea o dezvoltare a barei determinata strict. Imediat cum para date istorice ale unui timeframe mai mic, interpolarea fractala se va aplica acestor noi date. Cu toate acestea, sunt folosite 6 bare anterioare, nu 12. Adica sunt reproduse preturi reale ale Open, High, Low, Close plus inca 2 preturi generate. Valoarea si locatia acestor doua preturi generate depind de miscarea pretului in cele 6 timeframuri precedente.



Every tick – (fiecare miscare de pret) - (bazat pe toate disponibilele mai putin timeframes cu interpolare fractala pe fiecare miscare de pret)

Acest mod permite modelarea cea mai precisa a miscarilor de pret intr-o bara. Spre deosebire de „control-points”, metoda fiecare miscare de pret foloseste spre generare nu doar date ale celui mai apropiat timeframe mic, dar si cele ale ale tuturor timeframe-urilor mici disponibile. In acest fel, data exista mai multe date decat ale unui timeframe pentru un anumit interval de timp simultan, datele celui mai mic timeframe sunt utilizate pentru generare. Precum metoda precedente, aceasta metoda utilizeaza generare gractala a punctelor de control. Interpolarea fractala este folosita din nou pentru a genera miscare de pret intre punctele de control. Se poate intampla ca mai multe miscari de pret identice sa fie generate unul din celalalt. In acest caz, cotatiile duplicate sunt filtrate, si volumul ultimului dintre aceste cotatii succesive este fixat.



Trebuie luata in considerare un voluma mare al datelor miscarilor de pret generate. Aceasta poate influenta consumul sistemului de operare si viteza testarii.

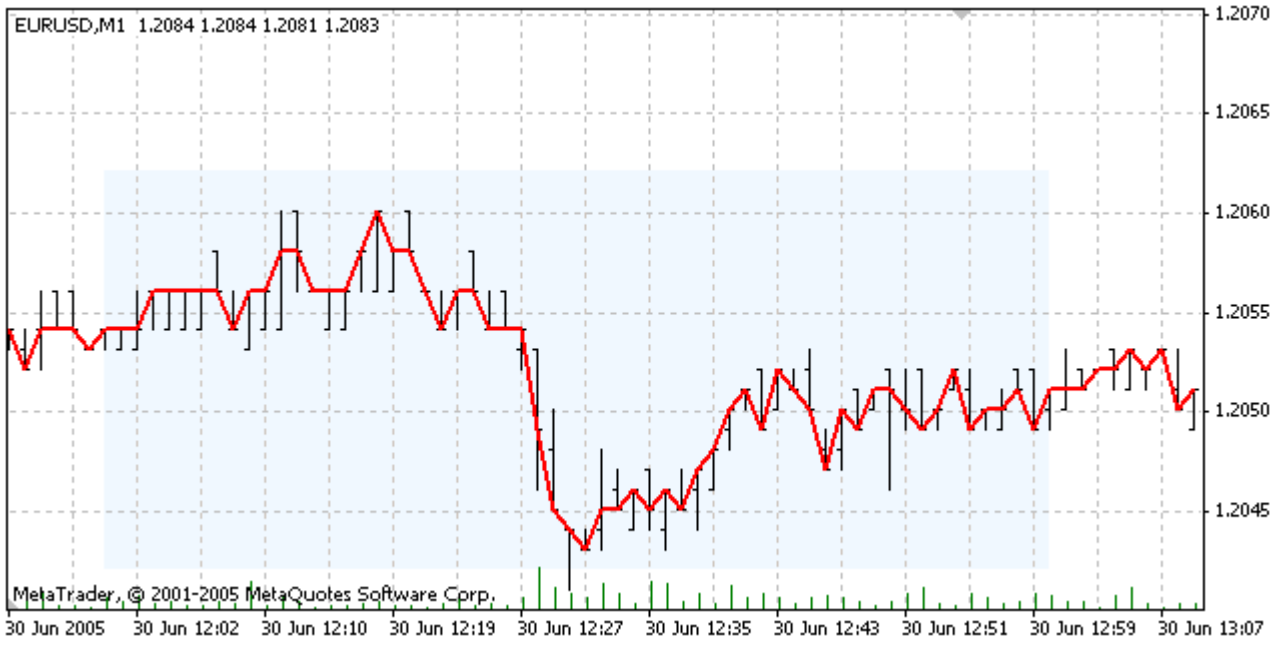
Atentie: Nu este necesara lansarea testarii tuturor miscarilor de pret daca nu exista timeframe-uri mai mici care sa acopere timeframe-ul testat. Aceasta modelare este destinata doar testarii pe baza datelor din timeframe-urile mai mici!

Utilizarea gamei de date in modelare

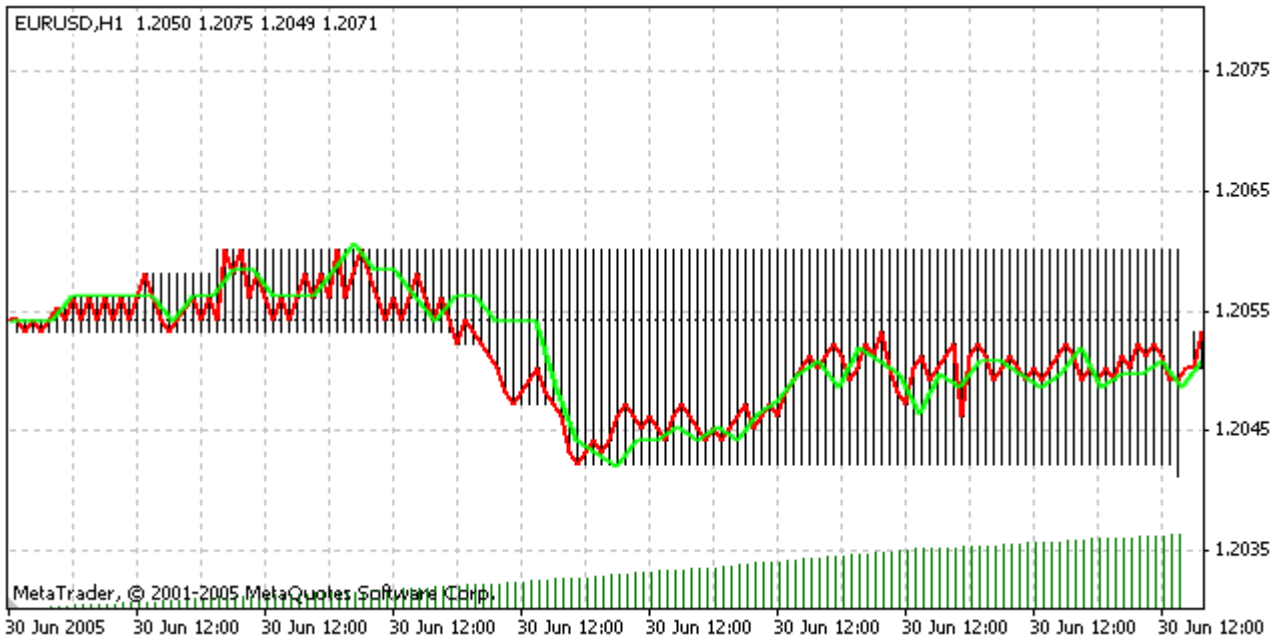
Gama de date poate fi setata in tab-ul Settings: bifati casuta „Use date” si specificati date in campurile „From:” si „To:”. Gama de date poate fi utilizata nu doar pentru testarea unui expert advisor, dar si pentru generarea de secvente de testare a barelor. Adesea nu este necesara generarea datelor pentru intreg istoricul, in special pentru modelarea Every Tick, unde volumul datelor neutilizate poate fi foarte mare. Asadar, daca la generarea initiala a secventei de testare (sau intotdeauna, daca este bifata casuta „Recalculate”) exista posibilitatea folosirii unei game de date, atunci barele, care intrec gama, nu au fost generate, doar suprascrise in secventa de rezultat. Datele nu sunt excluse din secventa pentru a permite posibilitatea calcularii corecte a indicatorilor pentru intreg istoricul primit. Trebuie mentionat ca primele 100 de bare nu sunt nici ele generate, aceasta limitare nu depinde de gama de date care se seteaza.

Referinta la Timeframe M1

Pentru a verifica acuratetea modelarii intrabar, vom utiliza graficele la minut din 30 Iunie 2005, de la 12:00 a.m. and 1:00 p.m.



O simpla scalare a graficului original la minut (linia verde este Close price) si a suprapunerii sale pentru graficul cu rezultatele din datele tuturor miscarilor de pret, rezulta practic intr-o coincidenta exacta:



Concluzii

Testari cu prezie maxima pot fi execute si veridicitatea simultana poate fi presupusa de asemenea daca exista timeframe-uri auxiliare suplimentare care sa acopere timeframe-urile testate 100%. Adica, problema testarii calitative a fiecarei miscari de pret se transforma in cautarea datelor istorice detaliate...

Testarea caracteristicilor si limitelor in MetaTrader 4

Introducere

Acest articol permite aflarea mai multor informatii despre caracteristicile si limitele Testarii de strategii in MetaTrader4.

Caracteristici speciale ale testarii strategiilor prin date istorice

- ***Unele functii sunt procesate/trecute fara output***
  Acestea sunt Sleep(), Alert(), SendMail(), SpeechText(), PlaySound(), MessageBox(), WindowFind(), WindowHandle(), WindowIsVisible()

- ***Tranzactionarea este permisa doar pentru simbolul testat, nu se poate testa portofoliul***
  Incercarea de a tranzactiona folosind un alt simbol se va reintoarce drept eroare

- ***Marimile loturilor care includ marimea initiala si pasul de tranzactionare, comisioanele si punctele swap trebuie luate din setarile contului activ***
  Inainte de testare, este necesar sa va asigurati ca este cel putin un cont activ in lista din fereastra „”Navigator” a terminalului.

- ***Toate punctele swap, cerintele de marja, expirarile, ordinele GTC sunt modelate***
  Testarea este efectuata in conditii cat se poate de apropiata de conditiile serverului de tranzactionare. Dar pot aparea niste inadvertente in estimarea cerintelor de marja pentru cotatiile indirecte din cauza lipsei de informatii precise despre preturile de conversie din fiecare moment.

- ***Bara zero a unui alt timeframe al aceluiasi simbol supus testului este modelata aproximativ***
  Open = correct Open, Close = correct Close, Low = min (Open,Close), High = max (Open,Close), Volume = final Volume (false)

- ***Modul Instant Execution este folosit in tranzactii, fiind procesat fara derapaje***

- ***Procesarea ordinelor Open/Close fara derapaje***

- ***Testarea se opreste dupa StopOut***

- ***Timeframe-urile saptamanale, lunare si neregulate nu sunt testate***

- ***Moneda de depozit poate fi schimbata, dar preturile de conversie sunt stabilite si cele curente disponibile sunt folosite***

- ***Inca nu exista intarzieri in executia operatiunilor de tranzactionare***
  Se planuieste sa se introduca un setup de intarziere in procesarea tranzactiilor

- ***Istoricul contului este disponibil in intregime si nu depinde de setari***

- ***Daca se folosesc alte simboluri si perioade in mod activ, este recomandabil downloadarea lor in profunzime***

- ***La modelarea every-tick, tester-ul pompeaza toti timeframe-ii necesari pentru simbolul testat in mod independent***

- ***Utilizarea functiei MarketInfo genereaza erori ERR_FUNCTION_NOT_ALLOWED_IN_TESTING_MODE(4059)***, cu toate acestea, sunt prezente informatii corecte despre preturile curente pentru simbolul testat, despre dimensiunile nivelului stop, despre marinea punctului, despre marimea spread-ului al oricarui simbol prezent in ferestrele cu cotatii.

Caracteristici speciale ale procesului de optimizare

- ***Nimic nu este rezultat in jurnal (altfel functia Print())***
  Aceasta a fost efectuata pentru a accelera testarea si a salva spatiu pe disc. Daca rezultatul reprezinta inregistrari complete, fisierele jurnalului vor avea nevoie de sute de mega byte-s.

- ***Desenarea obiectelor nu este setata cu adevarat***
  Obiectele sunt dezactivate pentru a accelera testarea.

- ***Este utilizata functia "Skip useless results"***
  Pentru a nu denatura tabelele si graficele cu rezultatele testate, este utilizata posibilitatea de a sari pentru rezultate foarte proaste. Aceasta functie poate fi activata in meniu, in tab-ul "Optimization Results" -> "Skip useless results".