visualchart

Visual Chart 6. Diseño de sistemas
3
Departamento de formación

www.visualchart.com

Diseño de sistemas con Visual Chart 6

CONTENIDO

1. Introducción a los sistemas automáticos.

¿Qué son los sistemas automáticos?

Llamaremos sistema al conjunto de reglas objetivas cuyo cumplimiento implica el envío de órdenes de compra y venta sobre un producto financiero. Cuando este conjunto de reglas se ejecutan sin la intervención de un ser humano a través de un robot o herramienta informática, entonces se considera que estamos ante un sistema de trading automático.

En Visual Chart, los sistemas automáticos nos permiten trasladar nuestras estrategias de trading al programa, facilitando la labor de operativa del inversor y **permitiendo eliminar el factor emocional**.

La finalidad de diseñar un sistema automático es la siguiente:

1. SIMULAR la operativa durante el histórico de datos.
2. VISUALIZAR los negocios generados aplicados a un gráfico.
3. Obtener los resultados ESTADISTICOS resultantes de la suma de todos los negocios.
4. OPTIMIZAR los parámetros variables de la estrategia.

Es decir, al elaborar un sistema automático, existe una fase previa durante la cual se analizará el funcionamiento de la estrategia durante los datos históricos. Este análisis permitirá informar al inversor si los resultados arrojados previamente son lo suficientemente fiables como para invertir en el futuro con dinero real. Por tanto, su principal función será la de indicar la buena o mala calidad de la estrategia sin necesidad de perder dinero.
Una vez se considera oportuno invertir en la estrategia, los sistemas automáticos en Visual Chart incluyen una opción a través de la cual se pueden asociar los negocios del sistema con órdenes reales vinculadas con una cuenta (lo que denominamos TRADING AUTOMÁTICO).

Creación de un sistema automático

Los sistemas automáticos consisten en un conjunto de cálculos aplicados a una serie de datos (representados en un gráfico de un producto financiero) que generan órdenes cuando se cumplen una serie de condiciones.
Dichas condiciones se definen mediante programación. En Visual Chart 6 esta programación se desarrolla a través de la Plataforma Visual o bien mediante los lenguajes de programación .NET.

El proceso de creación y funcionamiento de los sistemas sería el siguiente:

1. Se diseña el código con las reglas para una barra.
2. Se inserta el sistema sobre un gráfico.
3. Visual Chart comprueba sobre cada una de las barras del gráfico las condiciones diseñadas y genera los resultados.

Cuando se diseña un sistema, se establecen las reglas de compra y venta de forma genérica, y no pensando en un producto financiero en concreto. Para ello, existirán una serie de variables y funciones que hagan referencia a elementos comunes (precio, hora, fecha, volumen, etc…).

Al aplicar el sistema sobre un gráfico, estas variables y funciones sustituyen los elementos comunes por los datos específicos del gráfico en cuestión, de modo que a partir de dichos datos específicos, se verá si las reglas se van cumpliendo a lo largo de las barras del gráfico.

En el momento que un sistema queda asignado a un gráfico, visualizaremos las señales dadas sobre el mismo, y además, podemos llevar a cabo cada una de las funcionalidades del sistema (ver los resultados estadísticos, optimizar variables, activar el trading automático, etc…).

2. Los entornos de programación de Visual Chart 6.

En Visual Chart 6 tenemos la posibilidad de trabajar con sistemas de formas:

1. **Utilizando sistemas ya creados.** Bien sea porque forman parte de la lista de sistemas públicos de Visual Chart o bien porque son sistemas compartidos por otros usuarios.
2. **Diseñando nuevos sistemas.** En este caso, se trataría de sistemas que sigan nuestras propias ideas, lo que nos va a permitir conocer sobre qué productos funciona mejor nuestra estrategia.

La segunda opción podemos resolverla a través de dos posibles métodos de programación:

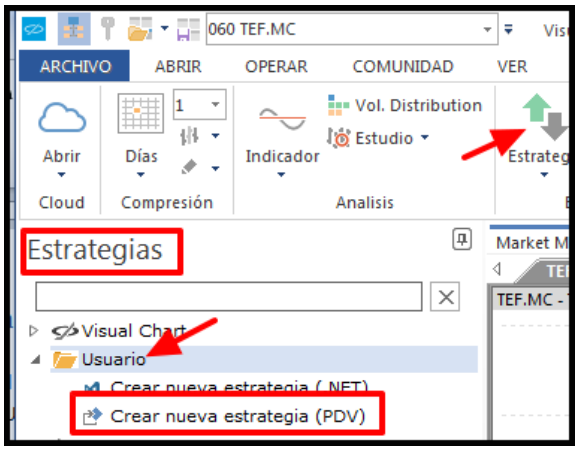
1. A través de la Plataforma Visual (PDV).
2. A través de la programación .NET.

La Plataforma de Diseño Visual (PDV)

La Plataforma de Diseño Visual (PDV) **es un asistente que pertenece a Visual Chart.** A través de él podemos diseñar sistemas, indicadores y estudios sin necesidad de tener conceptos de programación.
Al diseñar una estrategia en PDV, una vez ésta queda registrada, Visual Chart genera automáticamente el código correspondiente en VB.NET. Por tanto, la PDV funciona como un puente entre el usuario y la

programación .NET, facilitando la labor de diseño para aquellos usuarios que no dispongan de los suficientes conocimientos.

La creación de un nuevo sistema mediante PDV se lleva a cabo a través del siguiente comando:



Una vez especificado el nombre del sistema, se abrirá el entorno de programación de la PDV.

Dentro de dicho entorno, definiremos tres zonas como las principales de ésta interfaz:

1. El entorno de trabajo

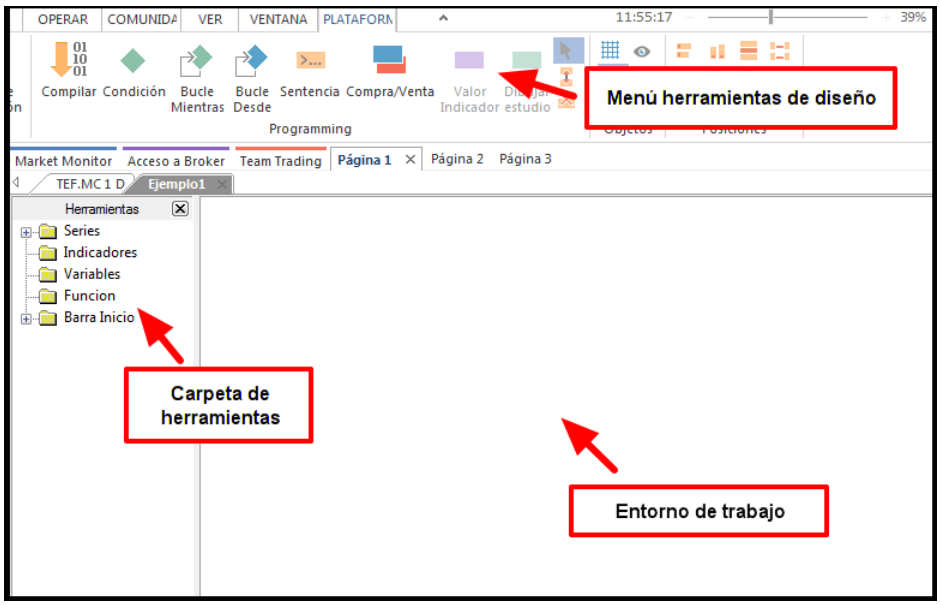
El entorno de trabajo es el espacio en blanco que aparece al abrirse la plataforma y es donde iremos dibujando el diagrama de flujo que representará al sistema.

2. Las carpetas de Herramientas

Estas carpetas contendrán a los distintos elementos que vamos a usar para montar el sistema. Más adelante veremos cómo añadir elementos a dichas carpetas.

3. El menú de Herramientas de diseño de programación

El menú de Herramientas contiene a las distintas herramientas que usaremos para diseñar el diagrama de flujo.



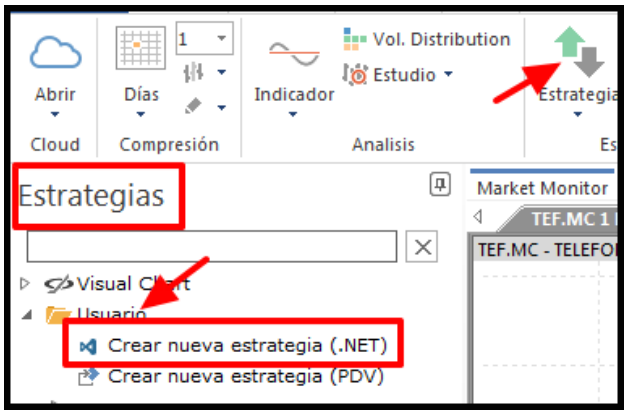
Más adelante entraremos en detalle acerca de cada una de estas zonas.

El editor de programación de Visual Studio (.NET)

Si tenemos nociones de programación o necesitamos desarrollar sistemas con cierta complejidad, disponemos de los lenguajes de programación .NET. Teniendo los conocimientos necesarios, esta opción es la más cómoda y versátil, ya que nos permite mayores posibilidades que la otra opción de programación (la Plataforma Visual).

El desarrollo de sistemas en .NET se hace a través del editor Visual Studio o bien Visual Studio Express. Esto dependerá de cómo haya configurado la instalación del programa el usuario.

La creación de un nuevo sistema mediante .NET se lleva a cabo a través del siguiente comando:



Una vez especificado el nombre del sistema, se abrirá Visual Studio (o Visual studio Express).

El editor se abrirá habiendo creado un proyecto con el nombre que le hayamos dado al sistema, incluyendo **una estructura preconfigurada** sobre la que debemos montar nuestro sistema. Visual Chart genera por defecto esta estructura para facilitar la labor al programador, de manera que sólo tenga que preocuparse de incluir los elementos vinculantes a su estrategia.

Como decimos, nos debemos centrar en las partes sobre las que tenemos que escribir, pudiendo obviar el resto de módulos. A continuación, veamos las características de cada una de dichas partes.

1. Zona Declaración de Variables

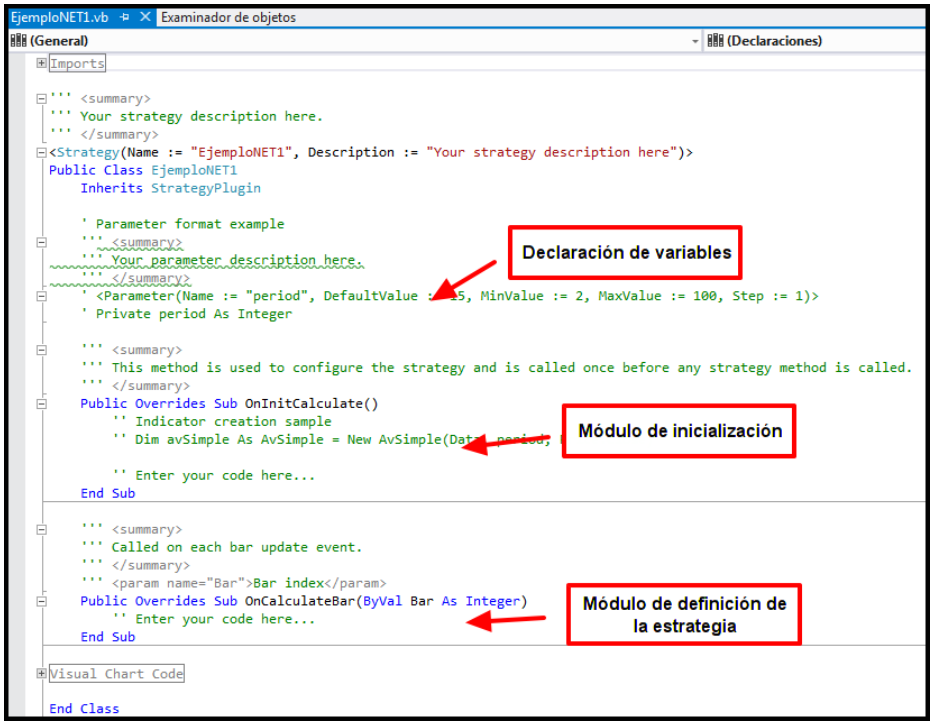
Dedicada a la declaración de variables que vamos a usar en la estrategia, diferenciando entre las que son parámetros del sistema, y después todas las que se vayan a manejar.

2. Procedimiento OnInitCalculate

Módulo desde donde inicializamos variables, se generan los objetos vinculados a los indicadores, se asignan valores constantes, etc... A este procedimiento recurrirá el programa sólo una vez, antes de comenzar los cálculos sobre las barras del gráfico.

3. Procedimiento OnCalculateBar

Módulo donde se define la estrategia: reglas de entrada y salida, operadores, etc... A este procedimiento recurrirá el programa una vez por barra, y siempre cuando dicha barra ha finalizado.



Más adelante profundizaremos en cómo se debe ir rellenando cada una de las partes.

3. Diseño de sistemas con la Plataforma Visual.

Comenzaremos la explicación del diseño de sistemas con la Plataforma Visual, ya que es la forma más adecuada de desarrollar una estrategia en caso de no tener nociones de programación. Más adelante veremos cómo diseñar sistemas en .NET para aquellos usuarios que sí posean ciertos conocimientos.

Puesto que el propósito principal es trasladar una idea de operativa a un lenguaje informático, lo primero que vamos a hacer es plantear precisamente dicha idea.

Idea de estrategia: Cruce del precio con una media

Las medias móviles permiten obtener el valor promedio de una serie de datos, lo cual sirve para establecer la tendencia general del producto: Cuando los valores de la media móvil suben, quiere decir que se está produciendo un crecimiento del activo. Si los valores de la media móvil bajan, entonces se estará produciendo una depreciación del valor.

En la siguiente imagen podemos ver la actuación de una media móvil de 30 sesiones aplicada al gráfico diario de SABADELL:

¿Cómo podemos aprovechar la información que aporta una media? Adelantándonos a la dirección de la media tanto en una dirección o en otra. Por ejemplo:

1. Compramos acciones si el precio actual está POR ENCIMA de la media. Si esto ocurre, quiere decir que el valor actual supera al valor promedio dado, de modo que puede ser señal de anticipación a una subida del precio.

2. Vendemos las acciones compradas si tras una subida el precio pasa a estar POR DEBAJO de la media. Nuevamente, tratamos de adelantarnos al próximo movimiento, en este caso a una caida, vendiendo lo más caro posible.

No obstante, esta idea tan mágica no siempre funciona: Hay ciertos momentos en los que el precio oscila en torno a su valor promedio, debido a que no hay una tendencia clara, lo cual supondrá pérdidas si seguimos la estrategia. Asumiendo esto, cabría ver si los resultados positivos permiten cubrir los momentos de pérdida como para que la estrategia sea rentable.

Para salir de esta duda, diseñaremos un sistema que nos devuelva los resultados históricos.

Empezaremos primero viendo cómo crearlo a través de la PDV. Antes de comenzar, cabe explicar cada una de las zonas que componen el entorno de programación, a fin de conocer mejor qué posibilidades ofrecen.

Zona Menú Herramientas de Diseño

Cuando abrimos el entorno PDV, vemos que en la cinta de opciones nos aparecen varios elementos. Estos elementos son las herramientas de diseño. De todas ellas, las que vamos a utilizar mayoritariamente son las siguientes:

1. Condiciones

Servirán para definir las reglas de entrada y salida u otras preguntas que queramos hacer desde el sistema.

2. Sentencias

Servirán para asignar valores a variables, como precios, resultados de fórmulas, etc...

3. Compra/Venta

Servirán para especificar el tipo de ordenes que queramos mandar en cada momento.

4. Compilar

Permite registrar el sistema en Visual Chart. Hasta que no pulsemos este botón, el sistema no estará disponible para su uso y tampoco estará guardado.

Zona Carpetas de Herramientas

Si seguimos nuestro ejemplo de la media cortada por el precio, veremos que lo que debemos hacer es añadir dos condiciones (una para la regla de compra y otra para la regla de venta) y añadir dos elementos de compra/venta. Eso es todo.

No obstante, primero debemos incluir en el sistema las herramientas que vayan a utilizarse.

Todas las herramientas deben incluirse dentro de las carpetas correspondientes al mismo tipo.

Veamos a continuación qué elementos podemos encontrar dentro de las carpetas de herramientas:

1. Series

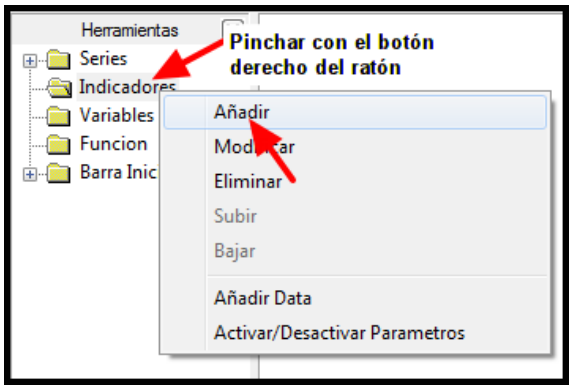
Las Fuentes o series son la base sobre la que se calculan los sistemas, indicadores y estudios. (datos históricos sobre una empresa, un índice, un Futuro etc.). Por defecto se incluye la serie Data1, que representará a la serie de datos sobre la que posteriormente aplicaremos el sistema.

La carpeta Series se incluye para aquellos casos en los que queramos trabajar con varias series de datos. En nuestro ejemplo esto no es necesario y por tanto obviaremos esta carpeta.

2. Indicadores

Si queremos usar la información de algún indicador en concreto, será necesario añadirlo. Para ello, pulsamos con el botón derecho sobre la carpeta Indicadores y seleccionamos la opción Añadir.

Seguidamente se abrirá un asistente para incluir indicadores.



3. Variables

Las variables son elementos cuyo valor puede variar a lo largo del diagrama de flujo. Se usan para almacenar distintas cosas, tales como parámetros del sistema, precios que queramos guardar, resultados de alguna función que calculemos, etc... El modo de añadir nuevas variables es similar al de los otros elementos, sólo que en este caso accedemos a la carpeta Variables.

Al incluir un indicador, si especificamos que sus parámetros serán parámetros del sistema, veremos que éstos aparecen en la carpeta de Variables.

4. Funciones

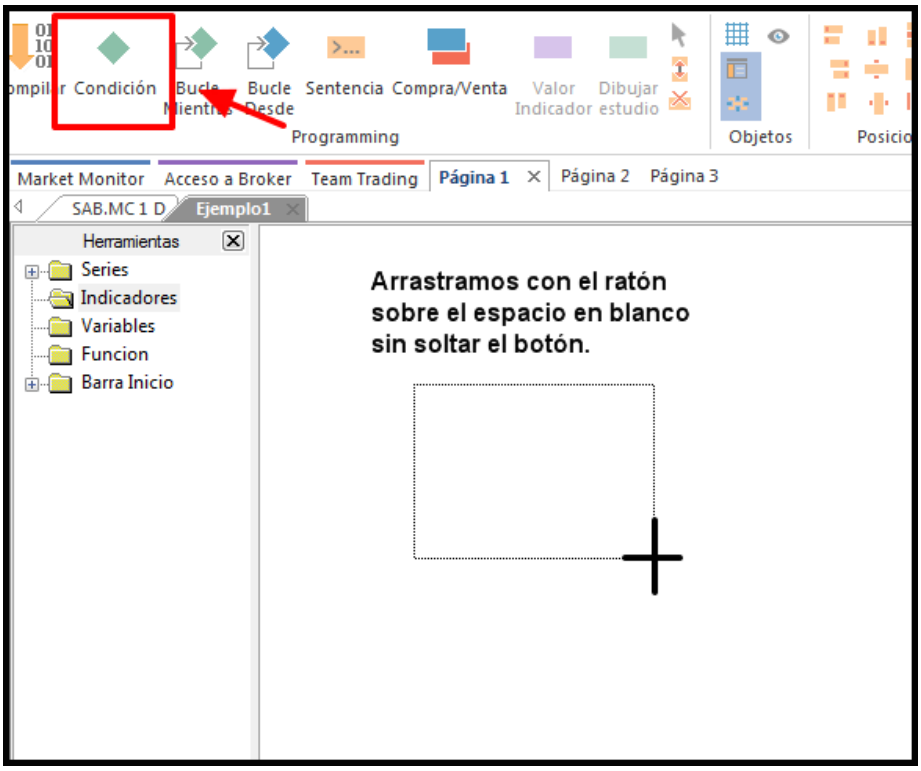
Las funciones poseen información acerca del estado del sistema, del valor de los precios, horario, fecha, etc... Las funciones relativas a la información de la barra (cierre, apertura, hora...) nos viene dada por defecto, sin embargo, el resto de funciones debemos incorporarlas al proyecto si queremos usarlas. Para ello, accedemos a la carpeta Funciones, pulsamos con el botón derecho y seleccionamos la opción Añadir.

En nuestro primer ejemplo vamos a añadir la función *GetMarketPosition*. Esta función nos dice si tenemos sobre la serie de datos posiciones abiertas o no.

Zona Entorno de Trabajo

Como ya hemos añadido a la media en el sistema y la función necesaria para saber si tenemos un negocio abierto, podemos proceder a montar el diagrama del sistema.

Como dijimos, tenemos que añadir una condición para preguntar si se cumplen las reglas de compra. Para ello, seleccionamos la herramienta de diseño Condición, y hacemos lo siguiente:



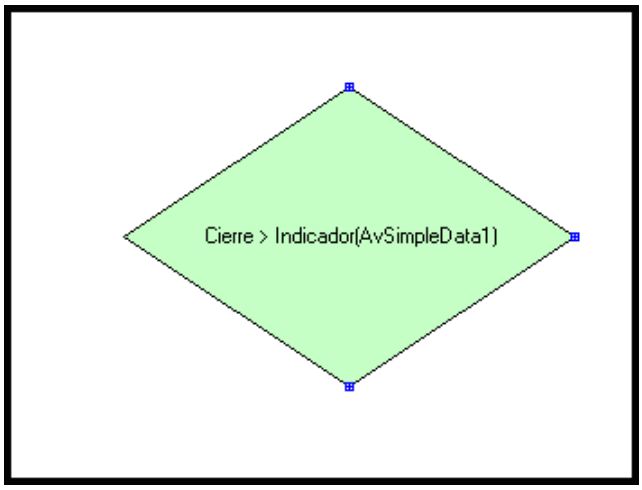
Al soltar el botón, se genera la condición y aparece el asistente para la creación de una nueva regla.

Todas las condiciones van a costar de dos elementos que se van a comparar y del símbolo de comparación (mayor, menor, igual, etc…).

En la lista de elementos (Identificador) encontramos los indicadores, series y funciones añadidas.

Al seleccionar Data1, podemos elegir entre los distintos campos asociados a una barra: Cierre, Máximo, Mínimo y Apertura de la barra. Al seleccionar el indicador, sólo podemos elegir la línea, puesto que el indicador no contiene más campos.

Nuestra regla de compra consiste en esperar a que el precio esté por encima de la media. Por tanto, el primer elemento será Data1 y el segundo el indicador. Como campo de la serie de datos elegiremos el Cierre, puesto que es el valor del activo en el momento de hacer la comprobación. El símbolo de comparación será “Mayor”. Al darle a Aceptar, se habrá generado la condición sobre el lienzo:



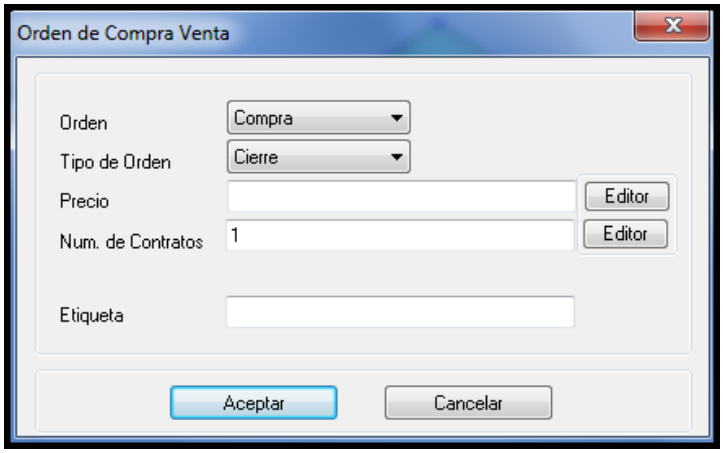
Este rombo verde equivale a la siguiente pregunta: ¿El precio de cierre de la barra actual es mayor estricto que el valor de la media en la barra actual?

Si la respuesta es correcta, el flujo del sistema continuará por la parte inferior del rombo.
Si la respuesta es incorrecta, el flujo del sistema continuará por el extremo derecho del rombo.

En nuestro caso, continuaremos el flujo por la parte inferior, es decir, cuando se confirme la señal.

Cuando ocurra esto, enviaremos una orden de Compra. El proceso para incluir un elemento de Compra/Venta es igual que en el caso de las condiciones. Por tanto, seleccionamos Compra/Venta y arrastramos sobre el lienzo.

Aparecerá el siguiente asistente:



En este asistente podemos especificar las diferentes características de la orden que vamos a enviar:

1. Orden

Permite escoger entre las siguientes opciones: Compra, Venta, Liquidar Largo, Liquidar Corto.

2. Tipo de Orden

Permite elegir una de las siguientes: Stop, Limitada, Cierre o Mercado.

3. Precio

Es el campo donde se indicará el precio al que se desea que se ejecute la orden. Puede ser un número sencillo, una expresión compleja o el valor de una variable. Para los 2 últimos casos comentados, será necesario pulsar sobre el botón Editor y se desplegará el cuadro de Crear sentencia, en el que indicará cualquier valor.

4. Número de Contratos

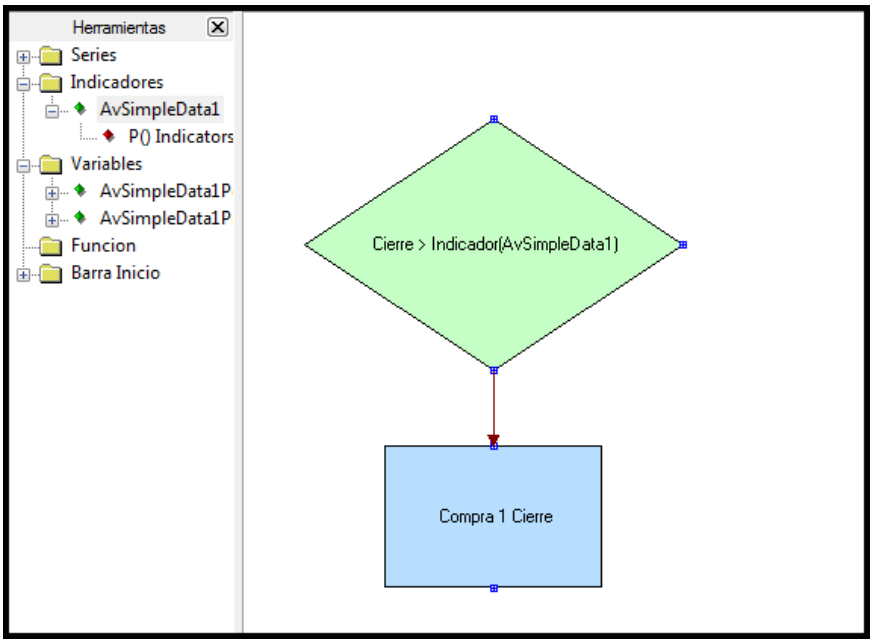
Es el campo donde expresaremos el número de contratos/acciones que deseamos comprar/vender. Puede ser un número sencillo, una expresión compleja o el valor de una variable. Para los 2 últimos casos, al pulsar sobre el botón Editor, se desplegará el cuadro Crear sentencia, en el que se indicará el valor deseado.

5. Etiqueta

Permite escribir un nombre con el que identificará a la orden que estamos estableciendo.

En nuestro ejemplo dejamos los valores que aparecen por defecto y le damos a Aceptar.

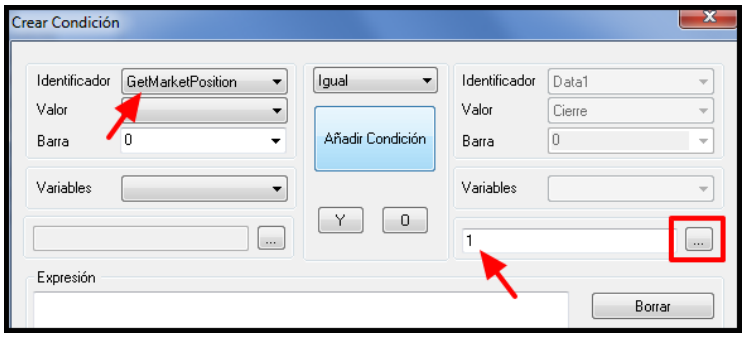
Por último, debemos unir la condición con la orden mediante una flecha para que el sistema sepa cómo queremos enlazarlos. El sistema quedará así:



Ya sólo nos quedaría especificar en qué casos debe vender las acciones/contratos comprados. Antes hemos dicho que si la respuesta a la condición era incorrecta, entonces el flujo del sistema salía por la derecha del elemento. A partir de este punto, enlazaremos una nueva condición donde vamos a preguntar si tenemos algún negocio abierto. No es necesario especificar si el precio está por debajo de la media, puesto que al ser la negación de que está por encima, se cumple directamente.

Añadimos por tanto una nueva condición en la que usaremos la función *GetMarketPosition*. El primer elemento será dicha función, el símbolo debe ser "Igual" y el segundo elemento debe ser el número 1.

Para poder escribir un 1, pinchamos en el botón (…) y escribimos un uno. La condición en el asistente quedará así:



Una vez añadida la nueva condición, enlazamos ambas condiciones por el nexo de la derecha.

Por último, incluimos un nuevo elemento de Compra/Venta, en el que lo que vamos a especificar es que queremos cerrar el negocio abierto.

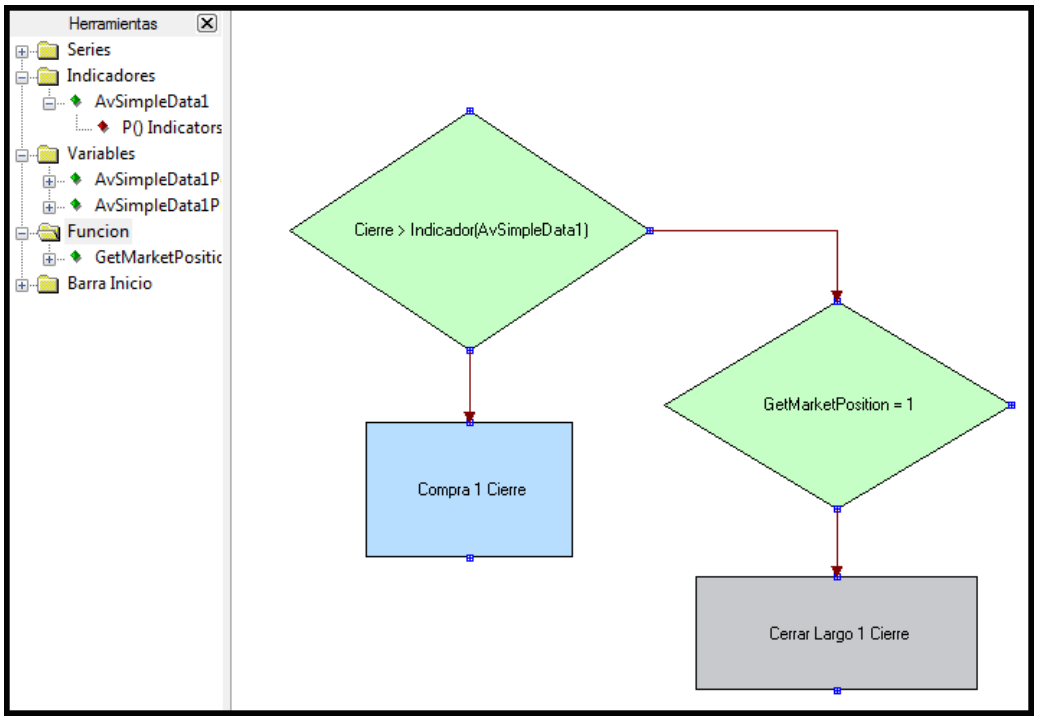
Visual Chart diferencia entre órdenes de entrada y órdenes de salida. Cada negocio está compuesto por estas dos órdenes, si bien se puede iniciar o entrando Comprados o entrando Cortos.

El concepto de entrar Cortos afecta sólo a algunos productos financieros y de momento no vamos a profundizar en ello: basta saber que cuando se especifica el modo de orden Venta, hace referencia a este tipo de entrada a Corto.

Por tanto, la orden de salida de un negocio de compra no será el modo de orden Venta, si no el modo de orden Cerrar Largo. Para abreviar, diremos que entrar Comprados y entrar Largos es equivalente, de ahí que a la orden de cierre de un negocio de compra se le llame Cerrar Largo.

Aclarado esto, en el asistente de Ordenes de Compra/Venta cambiaremos la Orden Compra por Orden Cerrar Largo. El resto lo dejamos igual y por último pulsaremos a Aceptar.

Nuevamente enlazamos la orden con la segunda condición a través de su nexo inferior. Con esto tendremos el sistema finalizado, el cual tendrá la siguiente apariencia:



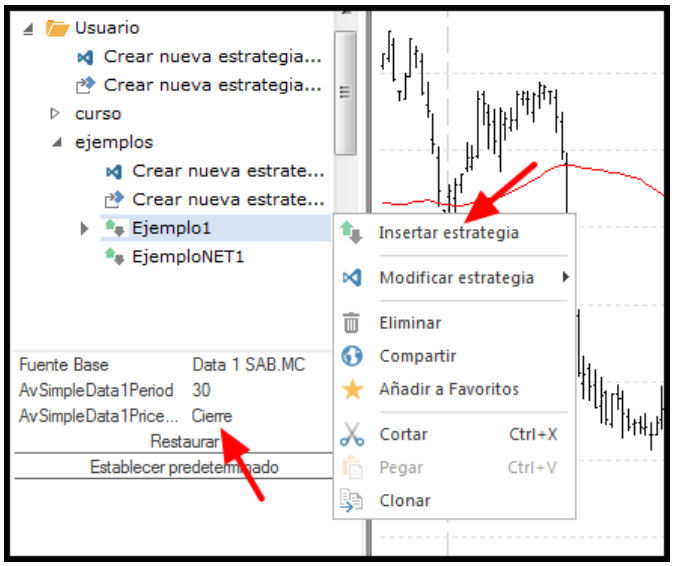
Acabamos el proceso de creación pulsando el botón Compilar de la cinta de opciones. En caso de que hubiera algún error, el programa nos avisará. En otro caso, el sistema se creará, se guardará y quedará registrado. Apareciendo a partir de ese momento dentro de la lista de sistemas de nuestro usuario.

Inserción del sistema sobre un gráfico

Una vez hemos diseñado el sistema, podemos cerrar el entorno de programación y volver al gráfico que habíamos abierto. En nuestro caso, se trataba del gráfico diario de SABADELL, sobre el que queríamos probar el sistema.

Para aplicar un sistema a un gráfico, seleccionamos en la cinta de opciones la opción Gráfico → Estrategia.

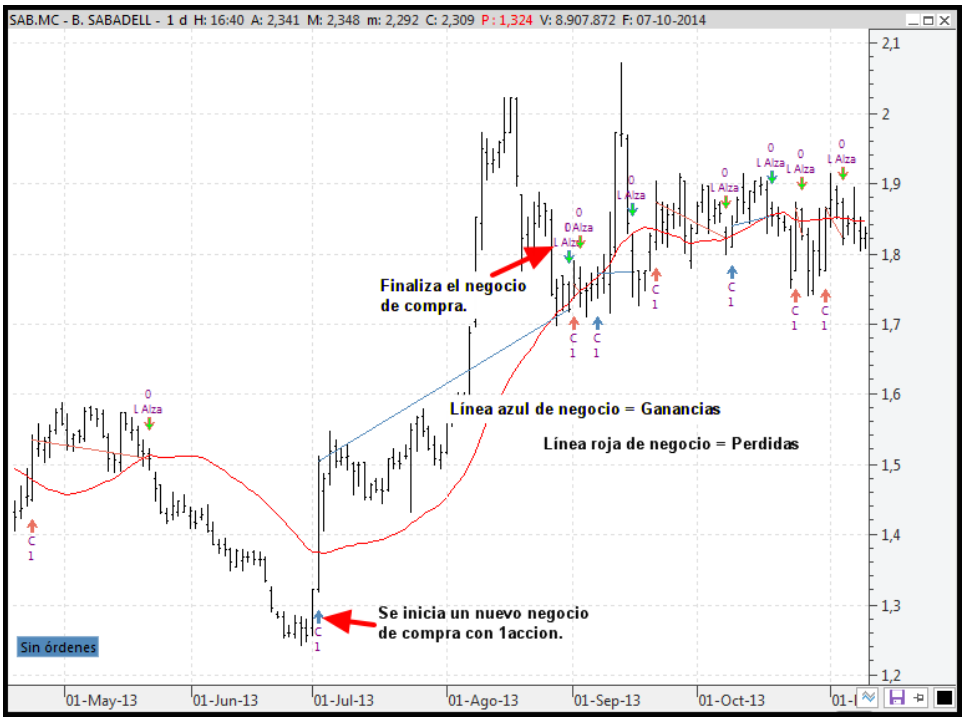
En la lista de sistemas que aparece a nuestra izquierda, buscamos el nuevo sistema creado. Si lo seleccionamos, nos aparecerá la opción *Insertar estrategia* y además, la lista de parámetros de la estrategia:



Si pinchamos directamente en *Insertar estrategia*, el sistema se calculará y aparecerán las señales sobre el gráfico.

Si antes accedemos a los parámetros y decidimos cambiarlos, podremos elegir otra combinación que no sea la predeterminada. Posteriormente podemos pulsar el botón Insertar Sistema para que se calcule sobre el gráfico.

Una vez aplicado, las señales que vemos nos indican lo siguiente:



4. Diseño de sistemas mediante .NET.

Si tenemos conocimientos de programación, la forma más cómoda de diseñar sistemas es a través del editor Visual Studio o Visual Studio Express. Visual Chart 6 permite la posibilidad de desarrollar estrategias mediante los lenguajes de programación .NET: C# y VB.NET.

Como decíamos al presentar este entorno de programación, al crear un nuevo sistema se genera un proyecto con una estructura base sobre la que trabajaremos. Además, dicho proyecto incorpora clases heredadas de Visual Chart para poder disponer de todos los elementos necesarios para el diseño de una estrategia.

Cuando diseñamos un sistema, necesitamos obtener información de distinto tipo para poder generar las reglas de entrada y salida, así como una serie de métodos con los que definir las operaciones a realizar.

Elementos principales

Veamos a continuación los elementos que principalmente utilizaremos:

1. Funciones

Las funciones poseen información acerca del estado del sistema, del valor de los precios, horario, fecha, etc... Las funciones son miembros de la clase APP, por tanto, para poder acceder a ellas, basta con teclear el punto y nos aparecerá la lista de funciones a las que tenemos acceso.

2. Indicadores

Si queremos usar la información de algún indicador en concreto, será necesario añadirlo siguiendo el siguiente procedimiento:

1. Declarar un objeto del mismo tipo que el indicador que queremos usar.
2. Crear/inicializar el objeto desde el método OnInitCalculate. Si el indicador lleva parámetros, interesa añadirlos como propios del sistema. Más adelante veremos cómo realizar la declaración de parámetros.
3. Una vez creado, podemos extraer el valor del indicador en una barra concreta mediante la función del indicador *Value()*.

3. Ordenes

Las órdenes se definen utilizando una serie de métodos concretos de la clase principal. Visual Chart distingue entre cuatro tipo de órdenes, por tanto, existe un método para cada una de ellas. Esta cuatro órdenes son:

1. Entrar a largo o compra = BUY
2. Entrar cortos o venta = SELL
3. Liquidar posición larga = EXITLONG
4. Liquidar posición corta = EXITSHORT

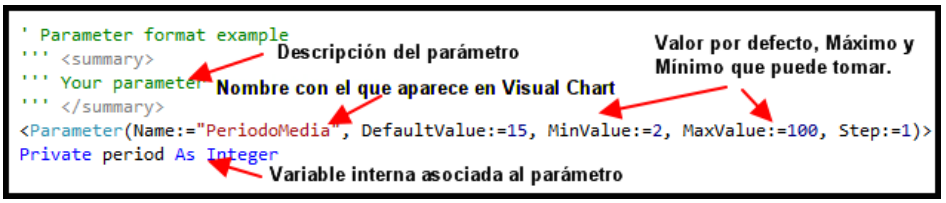
A la hora de la verdad, como es obvio, sólo estaremos hablando de operaciones de compra y venta, pero es necesario diferenciar entre operaciones para entrar y operaciones para salir puesto que Visual Chart las trata de distinta forma. A fin de ir directos a la explicación, como ejemplo, diseñaremos el mismo sistema que hemos visto anteriormente, es decir, un sistema que compre cuando el cierre esté por encima de una media y cierre posiciones cuando vuelva a cruzarla.

4. Variables

Las variables son elementos cuyo valor puede variar a lo largo del proceso. Se usan para almacenar distintas cosas, tales como parámetros del sistema, precios que queramos guardar, resultados de alguna función que calculemos, etc...

De entre ellos, caben destacar los parámetros del sistema, que deben de diferenciarse del resto de variables para que el programa sepa que se tratan de elementos diferentes. Para ello, se describen los atributos de cada parámetro.

La estructura para la creación de un parámetro, incluyendo sus atributos, sería la siguiente:
En VB.NET:



En C#:

```
/// <summary>
/// Periodo de la media simple.
/// </summary>
[Parameter(Name = "PeriodoMedia", DefaultValue = 15, MinValue = 2, MaxValue = 100, Step = 1)]
private int  period;
```

En este ejemplo, hemos añadido como parámetro del sistema el periodo de la media que usaremos.

Además, vamos a incluir como parámetros el precio de referencia de la barra para el cálculo de la media y además, vamos a incluir la cantidad de acciones a comprar con cada nuevo negocio.

El resultado sería el siguiente:

En VB.NET:

```
''' <summary>
''' Periodo de la media simple
''' </summary>
<Parameter(Name:="PeriodoMedia", DefaultValue:=15, MinValue:=2, MaxValue:=100, Step:=1)>
Private period As Integer

''' <summary>
''' Precio de referencia para el cálculo de la media
''' </summary>
<Parameter(Name:="PrecioMedia", DefaultValue:=Price.Close, MinValue:=Price.FirstPrice, _
    MaxValue:=Price.LastPrice, Step:=1)>
Private psource As Price

''' <summary>
''' Número de contratos/acciones por negocio.
''' </summary>
<Parameter(Name:="Contratos", DefaultValue:=10, MinValue:=1, MaxValue:=1000, Step:=10)>
Private contracts As Long
```

En C#:

```
/// <summary>
/// Periodo de la media simple.
/// </summary>
[Parameter(Name = "PeriodoMedia", DefaultValue = 15, MinValue = 2, MaxValue = 100, Step = 1)]
private int  period;

/// <summary>
/// Precio de referencia para el cálculo de la media.
/// </summary>
[Parameter(Name = "PrecioMedia", DefaultValue = Price.Close, MinValue = Price.FirstPrice,
    MaxValue = Price.LastPrice, Step = 1)]
private Price psource;

/// <summary>
/// Número de contratos/acciones por negocio.
/// </summary>
[Parameter(Name = "Contratos", DefaultValue = 10, MinValue = 1, MaxValue = 1000, Step = 10)]
private long contracts;
```

A la hora de trabajar con los parámetros dentro del código, haremos uso de las variables *period*, *psource* y *contracts*, que como vemos, su nombre no tiene por qué coincidir con el nombre asignado en los atributos del parámetro.

Declaración y creación de los indicadores

Siguiendo con el ejemplo, como vamos a utilizar una media simple, debemos añadir una nueva variable para hacer referencia a dicha media. En este caso, se tratará de un objeto de la clase *AvSimple*, que es, de hecho, el tipo de media a utilizar. La declaración de este objeto se haría de la siguiente manera:
En VB.NET:

```
''' <summary>
''' Objeto del tipo AvSimple que representa a la media a utilizar.
''' </summary>
''' <remarks></remarks>
Private avdata As AvSimple

''' <summary>
''' This method is used to configure the strategy and is called once before any str
''' </summary>
Public Overrides Sub OnInitCalculate()
    '' Indicator creation sample
    '' Dim avSimple As AvSimple = New AvSimple(Data, period, Price.Close)

    '' Enter your code here...
End Sub
```

En C#:

```
/// <summary>
/// Objetivo del tipo AvSimple que representa a la media a utilizar.
/// </summary>
private AvSimple avdata;

/// <summary>
/// This method is used to configure the strategy and is called once before any strategy
/// </summary>
public override void OnInitCalculate()
{
    // Indicator creation sample
    // AvSimple avSimple = new AvSimple(this.Data, period, Price.Close);
```

La declaración de objetos asociados a indicadores siempre se realiza antes del método *OnInitCalculate()*, así como la de otro tipo de objetos y variables que vayamos a usar en el sistema (salvo variables locales, obviamente).

La creación del objeto de la clase *AvSimple* la realizamos desde el método *OnInitCalculate()*.

Como dijimos, el programa accede a éste método al inicio del cálculo del sistema sobre la serie de datos. En ese momento es cuando se crean los objetos y se inicializan las variables globales:

En VB.NET:

```
''' This method is used to configure the strategy and is called once b
''' </summary>
Public Overrides Sub OnInitCalculate()
    Me.avdata = New AvSimple(Data, Me.period, Me.psource)
End Sub
```

En C#:

```
/// <summary>
/// This method is used to configure the strategy and is called once be
/// </summary>
public override void OnInitCalculate()
{
    this.avdata = new AvSimple(this.Data, this.period, this.psource);
}
```

La clase *AvSimple* incluye tres argumentos: Un objeto de clase *DataSeries*, un valor numérico que representa el periodo y una variable de tipo *Price*.

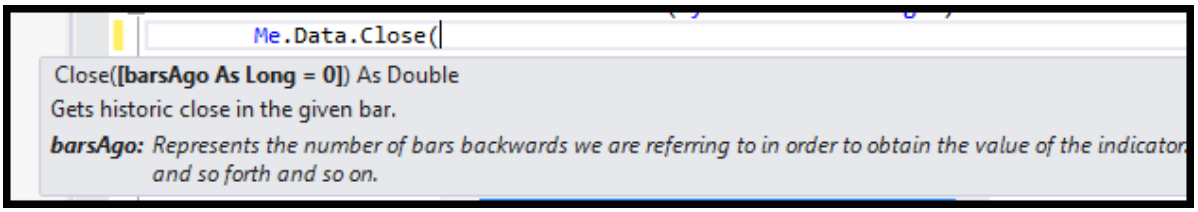
Cada vez que creamos una nueva instancia de objetos del tipo indicadores, nos van a aparecer los argumentos específicos de la clase de objeto. Sin embargo, en todos los casos, el primer argumento siempre será un *DataSeries*, puesto que en base a esto se especifica sobre qué serie de datos se debe de calcular el indicador en cuestión. Por defecto, siempre se especifica *Data*, el cual hace referencia a la serie de datos sobre la que posteriormente apliquemos el sistema.

Diseño de las reglas de operativa

Las reglas que decidirán en qué circunstancias el sistema comprará o venderá se especifican en el método *OnCalculateBar()*. Recordemos que el sistema accede a éste método cada vez que finaliza una nueva barra de la serie de datos.

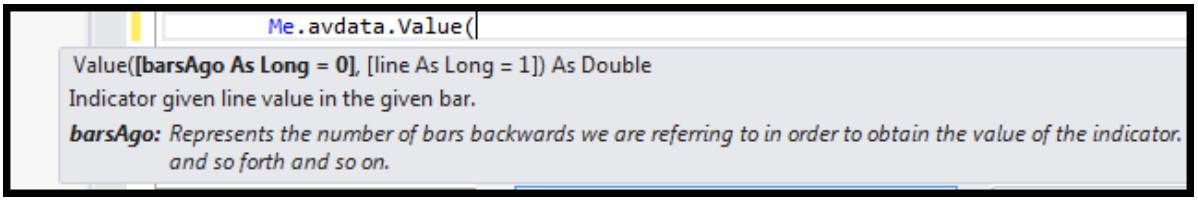
A efectos de posicionamiento, en cada momento, la barra sobre la que se está actuando ocupa la posición 0. La barra anterior a ésta, ocupa la posición 1. La siguiente, ocupa la posición 2, y así constantemente. El posicionamiento de las barras es importante puesto que la mayoría de los métodos y propiedades incluyen como argumento una posición de barra.

Por ejemplo. De cada barra podemos extraer cuatro precios concretos: Precio de apertura, precio de cierre, precio máximo y precio mínimo. Si quisiéramos saber el precio de cierre de la barra sobre la que estamos actuando en cada momento, haríamos uso de la función *Me.Data.Close():*



Esta función contiene el argumento *barsAgo*. Este argumento hace referencia a la posición de la barra. El valor dado por defecto es 0, es decir, la barra actual. Si en lugar de 0 escribiésemos un 1, extraeríamos el precio de cierre de la barra anterior a la actual. Y así constantemente.

Esta causística se repite con otras propiedades y métodos. Por ejemplo, la función *Value()* de un objeto de clase indicador, como puede ser el objeto de la clase *AvSimple* creado anteriormente, también incluye el argumento *barsAgo*:



Nuevamente, el valor dado por defecto es 0, es decir, el valor de la media en la barra actual. Aclarar en este caso que se trataría del valor de la media en el momento de cierre de dicha barra.

Volviendo al sistema, tenemos dos condiciones: La primera, que el precio sea mayor que la media para poder comprar. Y la segunda, que el precio sea menor y que tengamos posiciones abiertas para liquidar el negocio.

Crearíamos por tanto una sentencia *If..Else* utilizando los elementos mostrados anteriormente.

La regla de entrada sería así:

En VB.NET:

```
''' <summary>
''' Called on each bar update event.
''' </summary>
''' <param name="Bar">Bar index</param>
Public Overrides Sub OnCalculateBar(ByVal Bar As Integer)
    If (Me.Data.Close(0) > Me.avdata.Value(0)) Then
        Me.Buy(TradeType.AtClose)
    End If
End Sub
```

En C#:

```
/// <param name="bar">Bar index.</param>
public override void OnCalculateBar(int bar)
{
    if (this.Data.Close(0) > this.avdata.Value(0))
    {
        this.Buy(TradeType.AtClose);
    }
}
```

El método *Buy* cuenta con varios argumentos, si bien se pueden dejar los valores por defecto y especificar como tipo de entrada *TradeType.AtClose*. Este tipo de entrada quiere decir que la orden se enviará al precio de cierre de la barra donde se cumpla la condición.

Como hemos incluido en la lista de parámetros del sistema la opción de poder modificar la cantidad de contratos/acciones por negocio, vamos a incluir dicho parámetro dentro de los argumentos de la orden.

El argumento en cuestión es *contracts*, que coincide con el nombre que le hemos dado al parámetro, de modo que cambiaríamos la orden quedando de la siguiente manera:

En VB.NET:

```
''' <param name="Bar">Bar index</param>
Public Overrides Sub OnCalculateBar(ByVal Bar As Integer)
    If (Me.Data.Close(0) > Me.avdata.Value(0)) Then
        Me.Buy(TradeType.AtClose, contracts:=Me.contracts)
    End If
End Sub
```

En C#:

```
public override void OnCalculateBar(int bar)
{
    if (this.Data.Close(0) > this.avdata.Value(0))
    {
        this.Buy(TradeType.AtClose, contracts: this.contracts);
    }
    else
```

Por último, incluimos la condición para la liquidación del negocio. Para ello, usamos la sentencia *Else*, puesto que queremos liquidar cuando no se cumpla que el cierre esté por encima de la media.

Además, queremos comprobar que tenemos algún negocio abierto en el momento de enviar la orden.

Para ello, usamos la función *GetMarketPosition.* La condición quedaría de la siguiente forma:

En VB.NET:

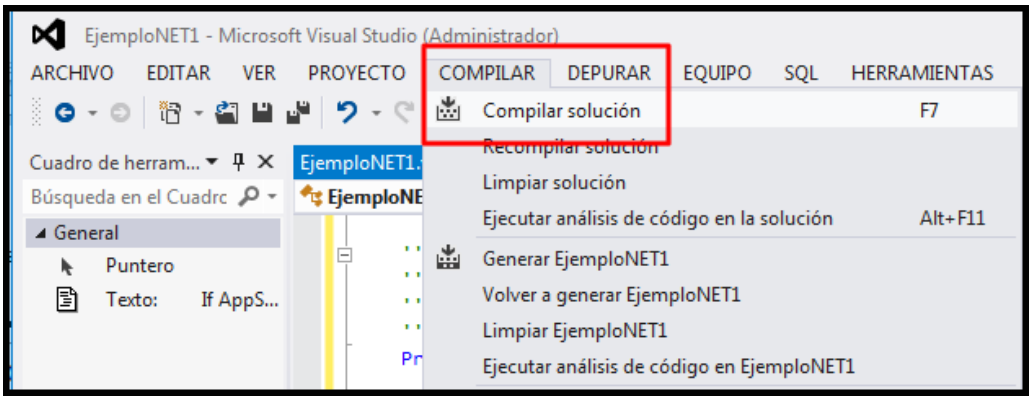
```
''' </summary>
''' <param name="Bar">Bar index</param>
Public Overrides Sub OnCalculateBar(ByVal Bar As Integer)
    If (Me.Data.Close(0) > Me.avdata.Value(0)) Then
        Me.Buy(TradeType.AtClose, contracts:=Me.contracts)
    Else
        If (Me.GetMarketPosition() = 1) Then
            Me.ExitLong(TradeType.AtClose, contracts:=Me.contracts)
        End If
    End If
End Sub
```

En C#:

```
public override void OnCalculateBar(int bar)
{
    if (this.Data.Close(0) > this.avdata.Value(0))
    {
        this.Buy(TradeType.AtClose, contracts: this.contracts);
    }
    else
    {
        if (this.GetMarketPosition() == 1)
        {
            this.ExitLong(TradeType.AtClose, contracts: this.contracts);
        }
    }
}
```

En la imagen vemos que los argumentos para el método *ExitLong()* son los mismos que para el otro tipo de orden. En realidad, todos los métodos asociados a órdenes incluyen los mismos argumentos.

Una vez diseñado el sistema, quedaría compilarlo y registrarlo. Esta acción se realiza desde el menú Compilar y despúes Compilar Solución, o bien pulsando el botón F7:



La inserción del sistema sobre un gráfico es exactamente igual que cuando el sistema está diseñado en PDV, ya que en lo que respecta al uso de sistemas no existen diferencias.

5. Aplicación de los sistemas

Una vez creado el sistema, podemos aplicarlo sobre cualquier serie de datos y en diferentes compresiones.

Para finalizar, repasaremos los distintos usos que le podemos dar a una estrategia automática en Visual Chart 6:

Simulación durante el histórico de datos

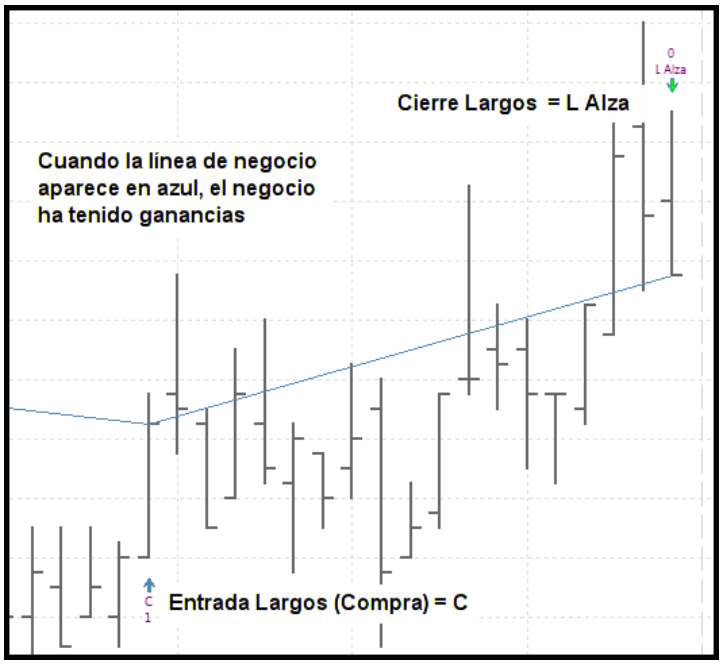
Cuando aplicamos un sistema a un gráfico, sobre éste quedan reflejadas las operaciones que ha realizado a lo largo del tiempo. Si ampliamos el histórico del gráfico, se recalculará nuevamente para las barras nuevas añadidas.

Asimismo, si dejamos correr el gráfico en tiempo real, veremos la generación de nuevas operaciones conforme se vayan cumpliendo las condiciones especificadas.

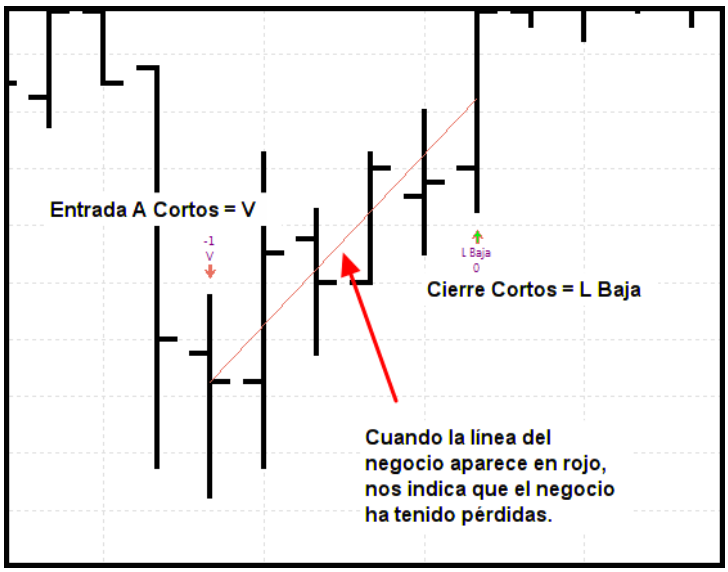
Si un sistema lleva incorporado para sus cálculos uno o varios indicadores, al aplicar el sistema al gráfico no veremos dibujados dichos indicadores sobre el mismo, de modo que si queremos ver la representación de estos para verificar las señales dadas, es necesario insertarlos uno a uno manualmente.

Las señales que nos podemos encontrar al aplicar un sistema son las siguientes:

1. Cuando la operación ha sido de Compra (Largos)



2. Cuando la operación ha sido de Venta (Cortos)

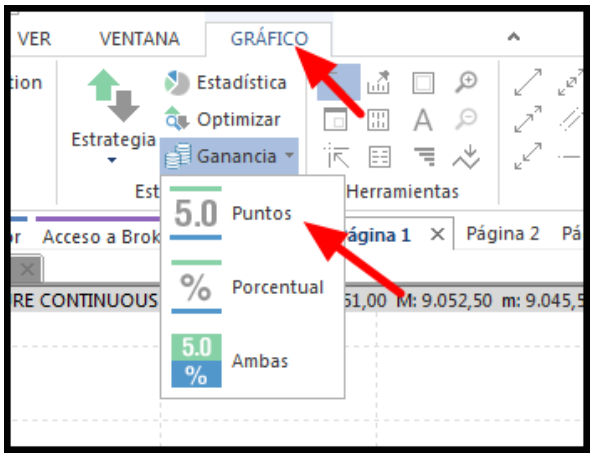


Extraer los resultados estadísticos

Puesto que en última instancia lo que nos interesa saber de un sistema es su rentabilidad, la posibilidad de extraer los resultados estadísticos del sistema se presenta como indispensable para el estudio de las estrategias automáticas.

Inicialmente, podemos observar sobre el gráfico, de forma directa, la línea de ganancia acumulada que ha ido obteniendo el sistema, tanto el resultado de la misma en puntos (valor monetario) como porcentual.

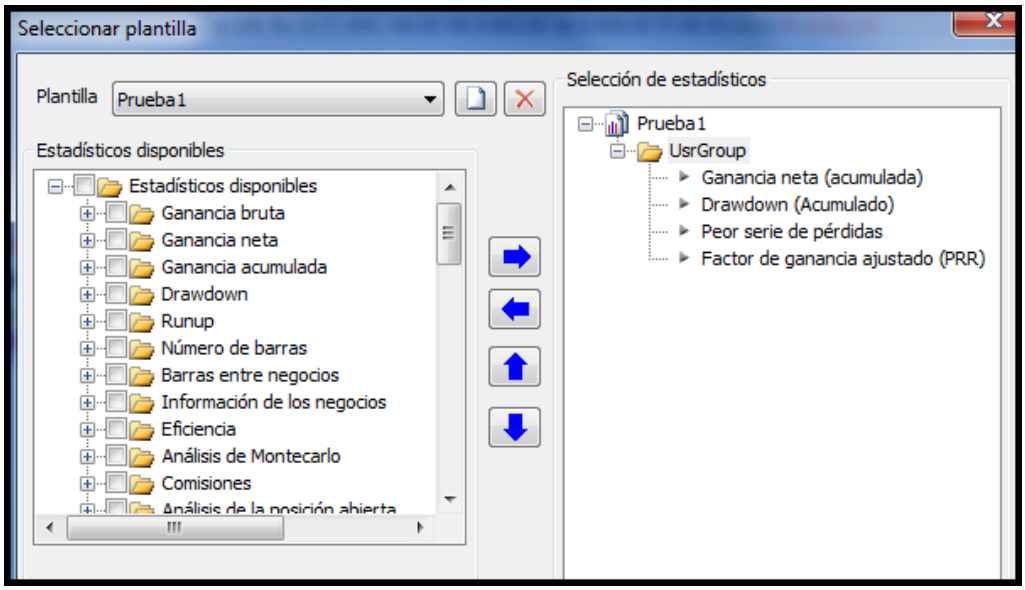
Para mostrar la línea de ganancia, seleccionamos en la cinta de opciones Gráfico y posteriormente Ganancia:



Al pulsar una de las tres opciones, automáticamente quedará dibujada debajo del gráfico:

Además de esto, podemos extraer una gran variedad de datos estadísticos. Para ello, accedemos a la cinta de opciones nuevamente, seleccionamos Gráfico y por último Estadística.

Los datos estadísticos que podemos obtener de un sistema son enormemente variados: Distintos métodos de ganancia, ratios, fiabilidad, rentabilidad, datos de entrada y salida, drawndown, comisiones, etc…



De todos ellos, los más utilizados suelen ser:

1) **Ganancia Neta Acumulada o Ganancia Neta Anual.** La ganancia nos va a indicar si obtenemos beneficio y en qué cantidad. La anual es especialmente interesante porque nos da una información que no depende del momento concreto en el que nos encontremos.

2) **La Peor Serie de Pérdidas** permite al inversor conocer cuanto debe soportar de pérdida máxima seguida, información realmente importante sobre todo desde el punto de vista psicológico: Si un sistema pronostica una ganancia acumulada de

20.000€ anuales pero con una serie de pérdida máxima de 12.000€, lo que nos quiere decir el sistema es que hay una esperanza de 20.000 al año pero que a lo largo del mismo podemos llegar a acumular 12.000 euros de pérdida.

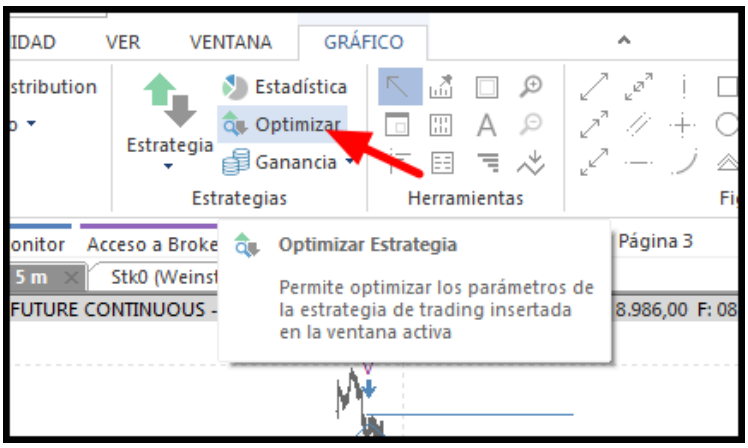
3) **El Factor de Ganancia Ajustado (PRR)** es uno de los ratios más utilizados para medir la efectividad de los sistemas. Este ratio tiene en cuenta tanto la ganancia anualizada como la peor serie de pérdidas, dando un valor que oscila entre 0 y 2. Si nuestro sistema devuelve un ratio inferior a 1, quiere decir que no es rentable y que al final nos generará pérdidas. Si su valor está por encima de 1,5, entonces estamos ante un sistema interesante y que puede aportar buenos resultados.

Optimización de los parámetros de la estrategia.

Ya hemos visto que un sistema puede contar con parámetros de entrada que permiten su versatilidad y adaptación a los diferentes productos financieros y a los cambios en el mercado.

Puesto que cada mercado tiene sus propias características, los valores que pueden funcionar en un mercado no tienen por qué hacerlo en otro. A fin de detectar qué combinación de parámetros actúa mejor en un producto concreto, Visual Chart 6 ofrece una potente herramienta con la cual podremos optimizar los parámetros de un sistema.

Esta herramienta es el **Team Trading**. Cuando pulsemos la opción Optimizar:



Se abrirá el menú del programa. Con el Team Trading no sólo vamos a poder sacarle el máximo provecho a nuestro sistema, sino que además contaremos con la opción de poder sacarle beneficios a través de la venta de la estrategia.