(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 April 2002 (25.04.2002)

PCT

(10) International Publication Number
WO 02/33636 A1

(51) International Patent Classification[7]: G06F 17/60

(21) International Application Number: PCT/US01/42517

(22) International Filing Date: 9 October 2001 (09.10.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/241,807	14 October 2000 (14.10.2000)	US
09/773,139	31 January 2001 (31.01.2001)	US

(71) Applicant *(for all designated States except US)*: **GOLDMAN, SACHS & CO.** [US/US]; 85 Broad Street, New York, NY 10004 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **OTERO, Hernan, G.** [AR/US]; 20 Briarcliff Pl., Huntington, NY 11743 (US). **HORN, Steven, B.** [US/US]; 315 E. 65th Street #7L, New York, NY 10021 (US). **TUMILTY, John** [GB/US]; 105 Duane Street, Apt. #22B, New York, NY 10007 (US).

(74) Agent: **CHOVANES, Joseph, E.**; Dilworth Paxson LLP, 3200 Mellon Bank Center, 1735 Market Street, Philadelphia, PA 19103 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 02/33636 A1

(54) Title: APPARATUS, METHODS AND ARTICLES OF MANUFACTURE FOR CONSTRUCTING AND EXECUTING COMPUTERIZED TRANSACTION PROCESSES AND PROGRAMS

(57) Abstract: Open-ended apparatus, methods and articles of manufacture for constructing and executing transaction processes and programs are shown. These apparatus, methods and articles of manufacture are primarily used in computerized trading processes.

APPARATUS, METHODS AND ARTICLES OF MANUFACTURE FOR CONSTRUCTING AND EXECUTING COMPUTERIZED TRANSACTION PROCESSES AND PROGRAMS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is related to provisional application U.S. Serial No. 60/241,807, by John A. Fanelli, Steven B. Horn, Hernan G. Otero and John Tumilty, which disclosure is incorporated herein by reference.

FIELD OF THE INVENTION

This invention relates to apparatus, methods and articles of manufacture for computerized transaction execution and processing. More particularly, this invention relates to apparatus, methods and articles of manufacture for client-server transaction execution and processing.

BACKGROUND OF THE INVENTION

Computerized transaction execution and processing requires an enormous, and often detrimental, amount of time and resources. The time and resources are required because, in most instances, execution and processing are based upon customized implementations of the transaction.

Customized transaction implementations require new programming. New programming requires cost and effort – not only for the first attempt, but also for the debugging and testing processes. Moreover, once the program is debugged and released, real world implementations require yet further testing and debugging.

All this effort takes resources and time. It takes resources because the programmers must first develop the program with input for the uses, and then the users themselves must test the program in the field, to ensure reliable operation. The effort required means that the users may be too busy doing their job to assist in programming efforts. Thus the program may not ever be developed. Moreover, by the time any particular program is developed, the markets may have shifted away from the initial transactional conditions that first provided the impetus for developing the program. For example, specific trading strategies are usually constructed and executed on a customized basis, yet by the time the program is developed for those strategies, and those strategies are executed, they may be no longer useful.

The cost, effort and time factors are not solely the result of required programming. In trading transactions, the programmers must be advised by the traders or other business professionals regarding desired trading strategies and desired markets. These professionals are busy in their own right – they may have little or no time to advise the programmers on what new strategies and markets should be developed. Even if they can advise the programmers, trading strategies can become quite complex, and in order to communicate those strategies and implement those strategies effectively, the programmer and trader interactions cost time, money and resources.

Enterprise-wide customization adds yet another level of time, effort and complexity. What may be useful in one enterprise business unit may not be useful in another, and time, effort and resources may not be available to implement specific programs customized for each business unit.

Finally, any implementations must be quite robust, and reliably and consistently execute trading strategies. The implementation of new computerized transactional programs must be as close to bullet proof as possible – failure of a trading programs can mean losses in thousands, millions or even billions of dollars. Developing reliable implementations of trading programs means that testing procedures and recovery procedures must always be paramount considerations.

Accordingly, it is an object of this invention to provide apparatus, methods and articles of manufacture for constructing and executing transactions.

It is a further object of this invention to provide open-ended apparatus, methods and articles of manufacture for constructing and executing transaction processes and programs.

It is a further object of this invention to provide robust and reliable apparatus, methods and articles of manufacture for implementing trading strategies.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a schematic diagram of a preferred embodiment.

Figure 2 is a schematic diagram of a preferred embodiment.

Figure 3 is a screen shot of a preferred embodiment.

Figure 4 is a screen shot of a preferred embodiment.

Figure 5 is a screen shot of a preferred embodiment.

Figure 6 is a schematic diagram of a preferred embodiment.

Figure 7 is a schematic diagram of a preferred embodiment.

Figure 8 is a flow chart of a preferred embodiment.

Figure 9 is a flow chart of a preferred embodiment.

SUMMARY OF THE INVENTION

The present invention provides apparatus, methods and articles of manufacture for open-ended construction and execution of computerized transaction processes. In the preferred embodiments, an engine is used that permits "plug-ins" to be used for construction, modification and alteration of trading procedure execution. These plug-ins can be pre constructed, or constructed when appropriate, and applied to the engine when desired.

In the preferred embodiments, the plug-ins comprise two types. The first type comprise algorithms used in trading. The second type comprise market-specific rules. Thus, for example, in the preferred embodiments, the engine can be configured with a specific algorithm and for a specific market for a first trade and then modified for another specific algorithm and another specific market for a second trade. In the especially preferred embodiments, the engine will carry out a number of trades using a specific algorithm, which has been chosen from a set of reconfigured algorithms. Moreover, the market plug-ins, having been set upon installation for use in a particular market, will be maintained for a predetermined or static period of time.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The preferred embodiments of the present invention provide apparatus, methods and articles of manufacture that have a number of characteristics in order to provide open-ended construction and execution of computerized trading processes. The preferred embodiments are constructed in Java which is essentially a platform independent language. Standard Java features are used in order to permit consistency among various Java versions. Use of Java permits convenient tracking of modifications and revisions of these embodiments, through Java features such as javadocs, introspection, reflection and the like. Of course, other embodiments may be

translated into other languages. Therefore, the embodiments may be used across a wide variety of networked platforms.

Figure 1 shows a schematic diagram of a preferred embodiment. At 10 is shown the engine infrastructure of the preferred embodiment. Written in Java, and present on the server, this software enables various data, plug-ins, applications, processes, and algorithms to be used in order to customized the trading process. These data, plug-ins, applications, processes, and algorithms are imported or plugged into the engine as desired in order to implement a particular trading strategy.

Seen in Figure 1 are various processes to be used in the engine 10. Area A of engine 10 symbolizes the area in which the plug-ins can be placed. Also seen at 10 is an area labeled "Market Specifics." This area, also supporting customization through data, plug-ins, applications, processes, and algorithms permits customization of any particular algorithm for any particular market in a manner explained in further detail below. In other embodiments, the plug-ins used for the various areas can be internal or external to the engine. Hereinafter, "plug- ins" will be used as a general term for data, plug-ins, applications, processes, and algorithms.

Engine 10, in this embodiment, provides services for the plug-ins. For example, most trading strategy plug-ins will need to access market data. Most trading strategy plug-ins will need to send orders to the exchange and be notified of executions, etc. Engine 10 provides these and other services to the plug-ins. For example in a preferred embodiment, engine 10 provides:

- A real time market data feed driver (e.g. Reuters SSL, TIB/Rendezvous feeds.)
- An exchange driver where the algorithm sends orders and receives executions back.
- A driver implementation that sends orders to one or more order management architecture(s) and/or system(s) server(s) is provided.
- An input driver which enters requests to the engine 10.

Figure 2 shows Process 1 implemented in engine 10. Process 1 might be a trading process such as Volume-Weighted-Average-Price or VWAP. The VWAP algorithm used in this embodiment, attempts to match the VWAP for a given instrument, such as an equity throughout a specified lifespan (e.g. throughout the full trading day). VWAP will maintain a number of limit orders in the market at different

price levels. In order to trade according to the VWAP algorithm of this embodiment, the engine will listen to market data throughout the day and access a volume profile to match the day's VWAP as close as possible.

The trader will then be able to review, thorough his screen, the order as it is being executed according to the VWAP algorithm. Any updates and/or changes will be simply made through his or her screen.

If a second VWAP algorithm was desired to be used, such as one that is based on theoretical values to trading, this second plug-in can be substituted for the first in the engine. This second plug-in will then be used by the engine.

Returning to Figure 2, the Market Specifics plug-in 1 has been chosen. Market specifics provide specific variables, data and other plug-ins necessary for the specific market in which the embodiment is being used. For example, they may be different limits on trading volume in one market versus another. The preferred embodiments permit configuration and modification of these Market Specifics, by plug-ins, so that the may be used in a variety of markets as desired.

In the preferred embodiments, the plug-ins comprise two types. The first type comprise algorithms used in trading. The second type comprise market-specific rules. Thus, for example, in the preferred embodiments, the engine can be configured with a specific algorithm, such as a first VWAP algorithm and for a specific market for a first trade such as the New York Stock Exchange and then modified for another specific algorithm such a Ratio algorithm and another specific market such as the Tokyo Stock Exchange for a second trade. In the especially preferred embodiments, the engine will carry out a number of trades using a specific algorithm, which has been chosen from a set of reconfigured algorithms. The algorithm used may be parameterized by the trader, in order to execute specific trades for a specific stock, price and number of shares. In these embodiments, the algorithm plug-in used is usually consistently used for that implementation of the embodiment during that particular trading period – whether it be an hour, day, week, etc. Of course, other embodiments may change their algorithm during any particular trading period. Moreover, the especially preferred embodiments usually maintain the market plug-in for at least the trading period, and usually longer. A trader, for example, may trade exclusively on the New York Stock Exchange using a preferred embodiment. Note that, using the especially preferred

embodiments, the trader will change the algorithm plug-in, embodying his or her trading strategy, much more frequently than his or her market plug-in, as he or she may only trade in a particular market. Network or enterprise wide implementations, however, will use the market plug-in order to configure any particular implementations for traders in the various trading markets.

This embodiment also effectively provides real-time monitoring of the order by the trader as well as others such as the sales force who desire to monitor the order and its execution. Additionally, orders are fully integrated, and so the trader or others may override individual orders through the system of this embodiment, without an additional messaging system. Similarly, any changes to an order, such as size of the order or a price limit or volume can be echoed to the system of this embodiment and the system will automatically adjust its trading to the new parameters.

Various screen shots of the administration and monitoring tool GUI (written in Java, using Swing) used in a preferred embodiment are shown at Figures 3 through 5. These are an Order Tracker screen shown in Figure 3, an Algorithm Configuration screen shown in Figure 4, and an Order Details screen shown in Figure 5. This tool allows for configuring algorithms as well as monitoring the server. This tool may be installed on either or both of the client and server machines and on more than one machine in the networked environment.

In the preferred embodiments, an algorithm is comprised of an Algorithm Context, which may be a Java Class, plus a set of event-action mappings. These algorithms are usually written by a programmer. The mappings may be modified by non-programmers (e.g. a trader) via the graphical tool. The mappings provide a powerful way to fine tune the algorithm. Of course other embodiments may modify the mappings in a different fashion. For example, the programmer may provide the trader or other end user with objects that constitute events, conditions and actions. The trader can then construct his or her own algorithms which are plugged into the invention in order to provide the trader with an automatic execution mechanism.

Other algorithms that may be used in this embodiment include:

> **Ratio** which tries to buy an instrument and sell a related instrument when the price between the two is more favorable than a specified ratio.

Gamma Hedge which hedges a portfolio and tries to capture volatility while doing so.

Aggressive Short Sell which tries to short sell a given instrument by making sure the Tokyo short sell rule is not violated.

Stop Loss which allows sending stop loss orders to exchanges that do not support this concept.

Iceberg which tries to trade a specified number of shares by sending only a part of the total order's quantity (the tip of the iceberg) to the market at any given time.

Auto Trader which decides whether to send trades to the market or fill from an account.

CB Delta Hedge which sends out underlyer market orders to hedge CB trades.

Of course, other algorithms or plug-ins may be used. Additionally, in the preferred embodiments, preferred methods of constructing and implementing new plug-ins are used. The preferred embodiments also use several Java features, e.g. introspection, reflection and the like, in order to automatically discover properties of the imported algorithms.

If new algorithms are desired, a number of methods can be used to create the algorithm. In this embodiment, if the new algorithm requires no Java code, then the algorithm can be created by leveraging on existing algorithm context classes. Specific classes have been established or predetermined in the preferred embodiments. If the new algorithm is simple enough, it can be created without writing any Java code, making use of the Administrator GUI. This can be done by simply creating a set of event-action mappings that will work on a pre-existing algorithm context class (e.g. the base AlgorithmContext class that is part of the preferred embodiments code classes).

Figures 6 and 7 show how various mappings or parts may be used to construct combinations. Those combinations, constructed in Figure 3, are then inserted into the

engine 20 in Figure 7. Note that a different Market Specifics plug-in, Market Specifics 2, has been chosen in Figure 7. These Market Specifics plug-ins may be from a predetermined set or constructed "on the fly." In the especially preferred embodiments, the market plug-in is usually maintained over some static trading period. A trader, for example, may trade exclusively on the New York Stock Exchange, using the market plug-in. In enterprise installations, the market plug-ins may be set for the particular trading markets across the enterprise, and remain as set for a predetermined or static period of time.

If the new algorithm requires writing new code, the fundamental classes within the architecture of the preferred embodiment are: AlgorithmContext, Action, ActionBindings, ActionDispatcher. New Actions might be needed, for new complex algorithms, in order to do simple tasks that the existing actions can not deal with. Algorithms which require saving state during the execution of the order, for example, need to have their own Algorithm Context subclass. The data will then be kept in this new subclass.

The following process is used in the preferred embodiment to place the code in a specific class. First, a meaningful Name and package must be chosen. Next, the Directory Structure must be created, according Java's requirement that classes be placed in a directory structure which corresponds to the package structure of the class. A Simple Algorithm Context must be written, starting with a template of what the class should look like, providing an empty, public constructor, adding in member variables, and providing a public getter/setter pair. Since this preferred embodiment makes use of beans support classes to access properties, JavaBeans conventions are used when naming these methods. Finally, in order to track the revisions, this embodiment uses a CVS Control System.

It is important to note that, in the preferred embodiments, traders provide vital feedback and oversight. Moreover, the embodiments evolve through use. There may be a lengthy tuning and feedback phase of algorithm development. The embodiments fit within a scalable architecture, and as the algorithms become more complex and widely used, the embodiments adapt and scale. Additionally, the embodiments must have fast Release Cycles. The preferred embodiments are flexible and separate the algorithm from the engine. Also, the algorithm should as orthogonal as possible to the

rest of the system. By use of this structure in the preferred embodiments, the embodiments can be used to trade and transact across virtually any instruments or exchanges.

In the preferred embodiments, the algorithms are tested for use. Of course, in other embodiments testing may not be desired. There are two main testing stages in a preferred embodiment. The first stage involves soliciting feedback with the traders and salespeople using the algorithm. The algorithm will not work right the first time, situations will not have been thought of, parameters will be wrong, failsafes will not be good enough and so on. The feedback at this early stage of development ensures not only a quick release but also that modifications can be made *in situ.*

The second stage of testing in this embodiment involves the continued evolution and updating of an algorithm once it is in production. It is important to have a very extensive series of tests that cover a multitude of trading situations. When changes are made to an algorithm, no matter how slight, every test is run and verified. This is necessary for production systems with a large number of users. Without high confidence that any changes made will not have any unforeseen follow-on effects, the release cycle becomes intolerably long. Of course, other embodiments may utilize different testing methods, including providing sample market feeds rather than real time feeds. The term "executing a trade" and its variants as used herein is meant to cover both actual and simulated execution of a trade.

The preferred embodiments implement a recovery mechanism, which assists the programmer in analyzing and/or recovering from crashes. The recovery process restores execution of orders by taking a number of steps. Those steps comprise:

> Recovering the state of the orders. This involves rebuilding the order hierarchy (parent/child relationships, executed quantities, etc.) as it existed prior to the crash.
>
> Recovering the exchange information. This involves making sure that all executions/corrections/cancellations that might have been pending when the embodiment crashed and had taken place during its blackout now get reflected in the embodiment's order hierarchy. This is done so

that future algorithm decisions get based on the current state of the world, and not the one present before the crash.

Restarting all algorithms. This is now possible since the algorithms will have their information up-to-date in order to make correct decisions on how to continue their execution. Depending on the complexity of the algorithms involved, this step may be as simple as setting up the event-action mappings for the algorithm context.

The recovery process in this embodiment includes writing to log or journal file. Of course other embodiments may have other recovery processes or recovery steps.

Figure 8 provides a flowchart summarizing processes of a preferred embodiment, from installation to trading. Figure 9 provides a flowchart summarizing a process for changing a plug-in. Other embodiments may have these processes or other processes with the same or similar steps in these or other orders.

The above description and the views and material depicted by the figures are for purposes of illustration only and are not intended to be, and should not be construed as, limitations on the invention.

Moreover, certain modifications or alternatives may suggest themselves to those skilled in the art upon reading of this specification, all of which are intended to be within the spirit and scope of the present invention as defined in the attached claims.

CLAIMS

We claim:

1. An apparatus for computerized trading comprising:
   - a first plug-in for implementing a trading strategy,
   - a second plug-in for implementing a trading strategy,
   - an engine for providing services to either of said first or second plug-in,
   - whereby said first plug-in is implemented in said engine in order to execute a trade.

2. An apparatus as in claim 1, wherein said second plug-in is implemented in said engine in order to execute said trade.

3. An apparatus as in claim 1, wherein said first plug-in further comprises an algorithm plug-in that is implemented by the engine.

4. An apparatus as in claim 1, wherein said second plug-in further comprises a market plug-in that is implemented by the engine.

5. An apparatus as in claim 1, wherein the first and second plug-ins, and the engine, are constructed in Java.

6. An apparatus as in claim 1, further comprising a third plug-in whereby said third plug-in is substituted for said first plug-in in said engine.

7. An apparatus as in claim 2, further comprising a fourth plug-in whereby said fourth plug-in is substituted for said second plug-in in said engine.

8. An apparatus for computerized trading comprising:

- a first, algorithm plug-in for implementing a trading strategy,

- a second, market plug-in for implementing a trading strategy,

- an engine for providing services to said first and second plug-ins, whereby said first and second plug-ins are implemented in said engine in order to execute a trade,

- a third algorithm plug-in,

- a fourth market plug-in,

whereby either of said third or fourth plug-ins may be substituted for either of said first plug-in or second plug-in respectively, in said engine, in order to execute a trade.

9. The apparatus of claim 8 wherein said first and third algorithm plug-ins implement trading strategies selected from a group comprising:

- Volume Weighted Average Price;
- Ratio;
- Gamma Hedge;
- Aggressive Short Sell;
- Iceberg;
- Auto Trader;
- CB Delta Hedge;
- Stop Loss; and
- Short Sell.

10. A method for computerized trading comprising:

- providing a first plug-in for implementing a trading strategy,
- providing a second plug-in for implementing a trading strategy,
- providing an engine for providing services to either of said first or second plug-ins, and,
- executing a trade using said first plug-in implemented in said engine.

11. A method as in claim 10, wherein the step of executing a trade using said first plug-in implemented in said engine further comprises the step of using said second plug-in implemented in said engine in order to execute said trade.

12. A method as in claim 10, wherein the step of using said first plug-in implemented in said engine further comprises using an algorithm plug-in.

13. A method as in claim 11, wherein the step of using said second plug-in implemented in said engine further comprises using a market plug-in.

14. A method as in claim 10, wherein the steps of providing a first plug-in for implementing a trading strategy, providing a second plug-in for implementing a trading strategy, and providing an engine for providing services to either of said first or second plug-ins, further comprises providing Java versions of said first and second plug-ins and said engine.

15. A method as in claim 10, further comprising the step of providing a third plug-in.

16. A method as in claim 15, further comprising the step of substituting said third plug-in for said first plug-in in said engine.

17. A method as in claim 10, further comprising the step of providing a fourth plug-in.

18. A method as in claim 17, further comprising the step of substituting said fourth plug-in for said second plug-in.

19. A method for computerized trading comprising:

- providing a first, algorithm plug-in for implementing a trading strategy,
- providing a second, market plug-in for implementing a trading strategy,
- providing an engine for providing services to either of said first or second plug-ins,
- implementing said first and second plug-ins in said engine,
- providing a third algorithm plug-in,
- providing a fourth market plug-in, and
- substituting either of said third or fourth plug-ins for either of said first plug-in or said second plug-in respectively, in said engine, in order to execute a trade.

20. A method as in claim 19, wherein the step of providing a first algorithm plug-in for implementing a trading strategy, further comprise providing a first algorithm plug-in selected from a group comprising:

- Volume Weighted Average Price;
- Ratio;
- Gamma Hedge;
- Aggressive Short Sell;
- Iceberg;
- Auto Trader;
- CB Delta Hedge;
- Stop Loss; and
- Short Sell.

21. A method as in claim 19, wherein the step of providing a third algorithm plug-in for implementing a trading strategy, further comprise providing a third algorithm plug-in selected from a group comprising:
    - Volume Weighted Average Price;
    - Ratio;
    - Gamma Hedge;
    - Aggressive Short Sell;
    - Iceberg;
    - Auto Trader;
    - CB Delta Hedge;
    - Stop Loss; and
    - Short Sell.

22. The method of claim 19, further comprising the step of initiating a recovery mechanism in the event of system failure.

23. An article for executing computerized trading comprising:
    - a computer – readable signal bearing medium;
    - means in the medium for providing a first plug-in for implementing a trading strategy,
    - means in the medium for providing a second plug-in for implementing a trading strategy,
    - means in the medium for providing an engine for providing services to either of said first or second plug-in, whereby said first plug-in is implemented in said engine in order to execute a trade.

24. An article as in claim 23, further comprising means in the medium for providing a third plug-in for implementing a trading strategy.

25. An article as in claim 24, further comprising means in the medium for substituting said third plug-in for said first plug-in in said engine.

26. An article as in claim 23, further comprising means in the medium for providing a fourth plug-in for implementing a trading strategy.

27. An article as in claim 24, further comprising means in the medium for substituting said fourth plug-in for said second plug-in in said engine.

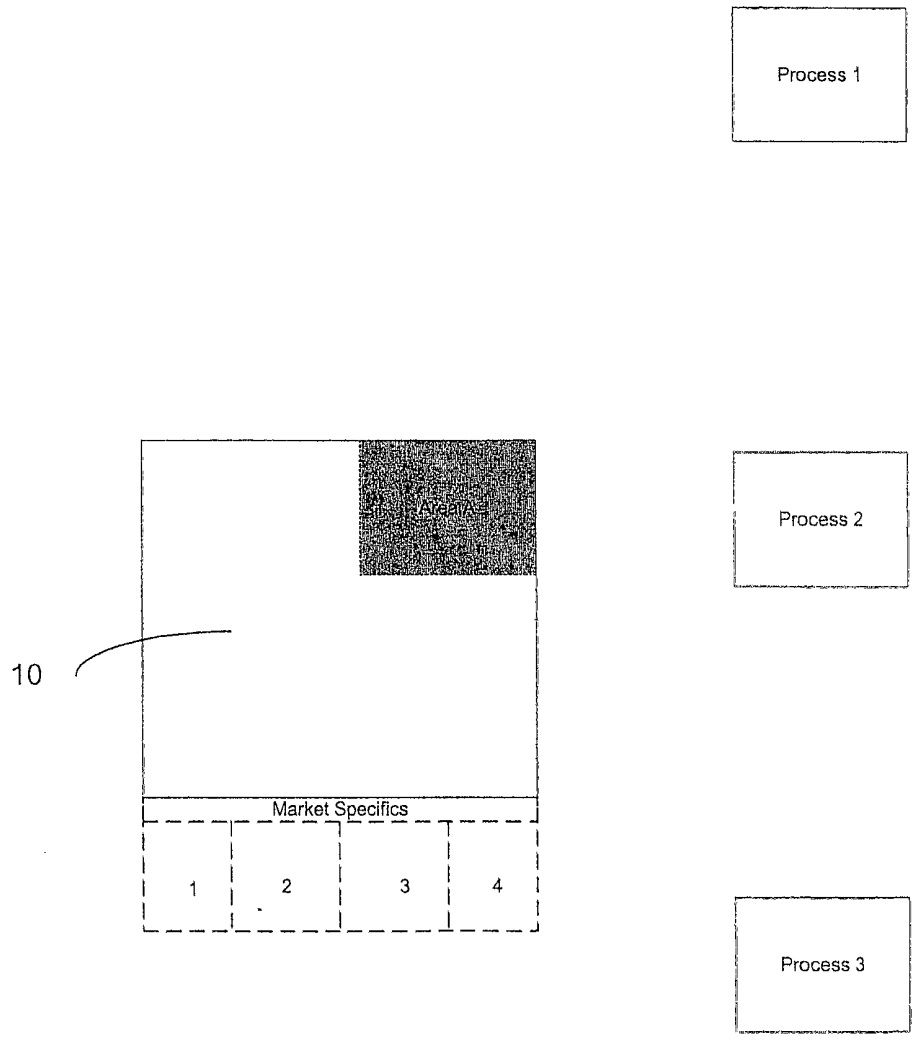


Figure 1

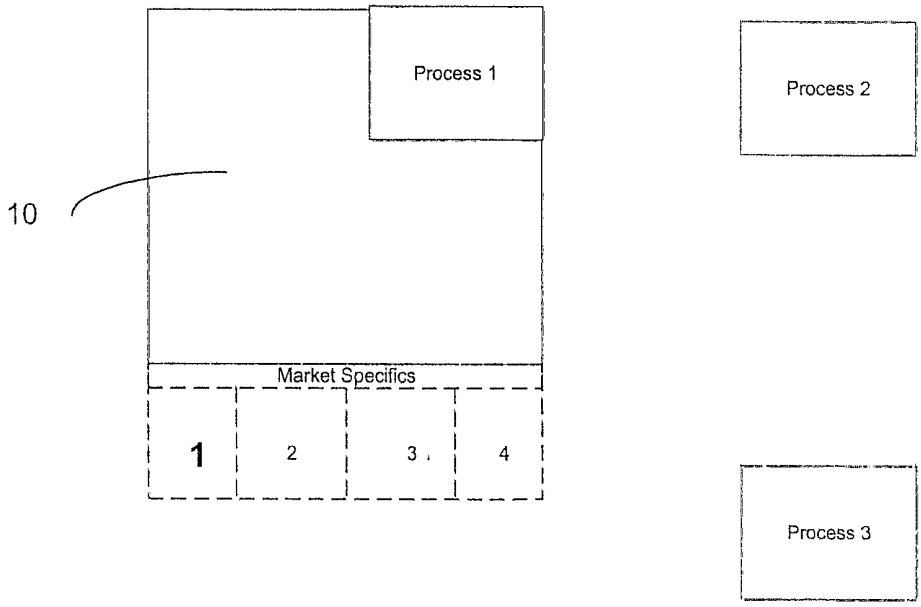
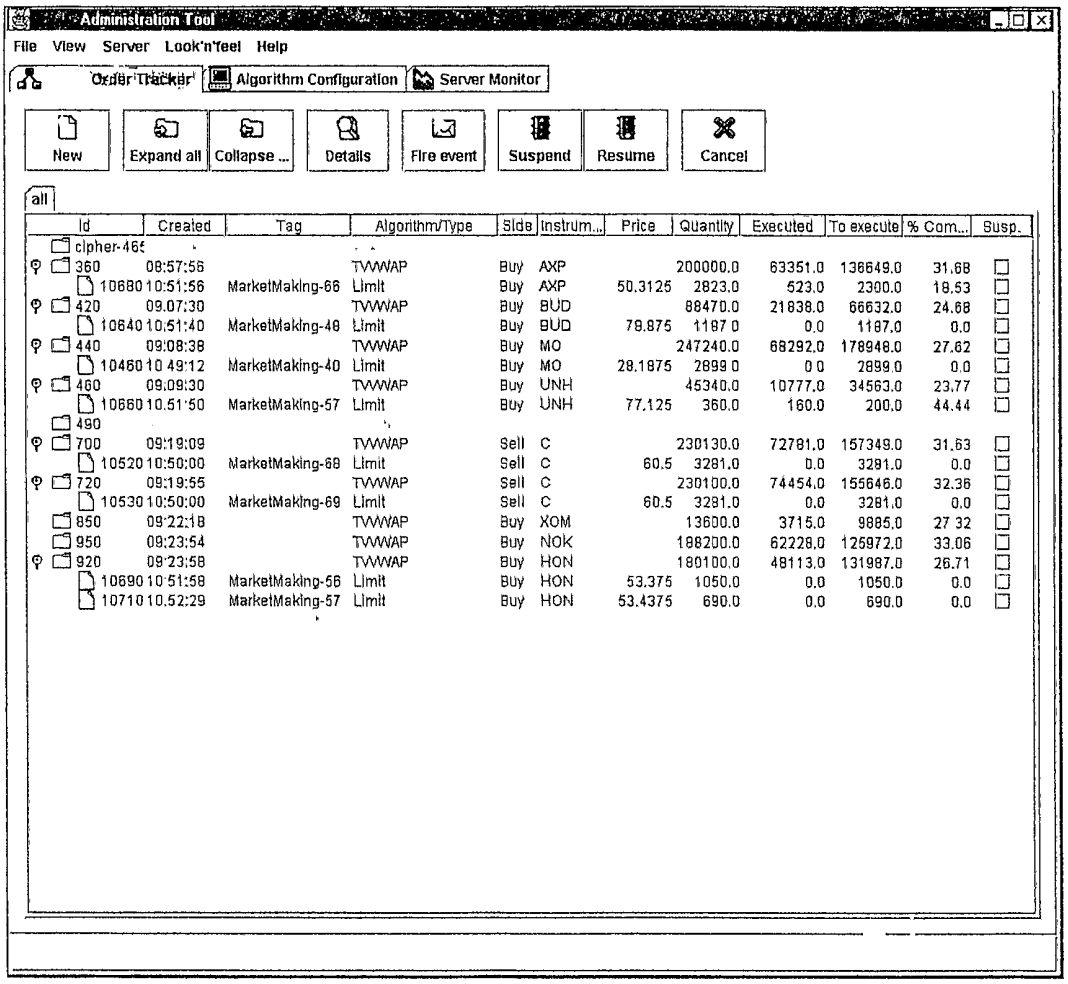


Figure 2



Id	Created	Tag	Algorithm/Type	Side	Instrum...	Price	Quantity	Executed	To execute	% Com...	Susp.
cipher-46											
360	08:57:56		TWWAP	Buy	AXP		200000.0	63351.0	136649.0	31.68	
10680	10:51:56	MarketMaking-66	Limit	Buy	AXP	50.3125	2823.0	523.0	2300.0	18.53	
420	09:07:30		TWWAP	Buy	BUD		88470.0	21838.0	66632.0	24.68	
10640	10:51:40	MarketMaking-48	Limit	Buy	BUD	78.875	1187.0	0.0	1187.0	0.0	
440	09:08:38		TWWAP	Buy	MO		247240.0	68292.0	178948.0	27.62	
10460	10:49:12	MarketMaking-40	Limit	Buy	MO	28.1875	2899.0	0.0	2899.0	0.0	
460	09:09:30		TWWAP	Buy	UNH		45340.0	10777.0	34563.0	23.77	
10660	10:51:50	MarketMaking-57	Limit	Buy	UNH	77.125	360.0	160.0	200.0	44.44	
490											
700	09:19:09		TWWAP	Sell	C		230130.0	72781.0	157349.0	31.63	
10520	10:50:00	MarketMaking-68	Limit	Sell	C	60.5	3281.0	0.0	3281.0	0.0	
720	09:19:55		TWWAP	Sell	C		230100.0	74454.0	155646.0	32.36	
10530	10:50:00	MarketMaking-69	Limit	Sell	C	60.5	3281.0	0.0	3281.0	0.0	
850	09:22:18		TWWAP	Buy	XOM		13600.0	3715.0	9885.0	27.32	
950	09:23:54		TWWAP	Buy	NOK		188200.0	62228.0	125972.0	33.06	
920	09:23:58		TWWAP	Buy	HON		180100.0	48113.0	131987.0	26.71	
10690	10:51:58	MarketMaking-56	Limit	Buy	HON	53.375	1050.0	0.0	1050.0	0.0	
10710	10:52:29	MarketMaking-57	Limit	Buy	HON	53.4375	690.0	0.0	690.0	0.0	

Figure 3

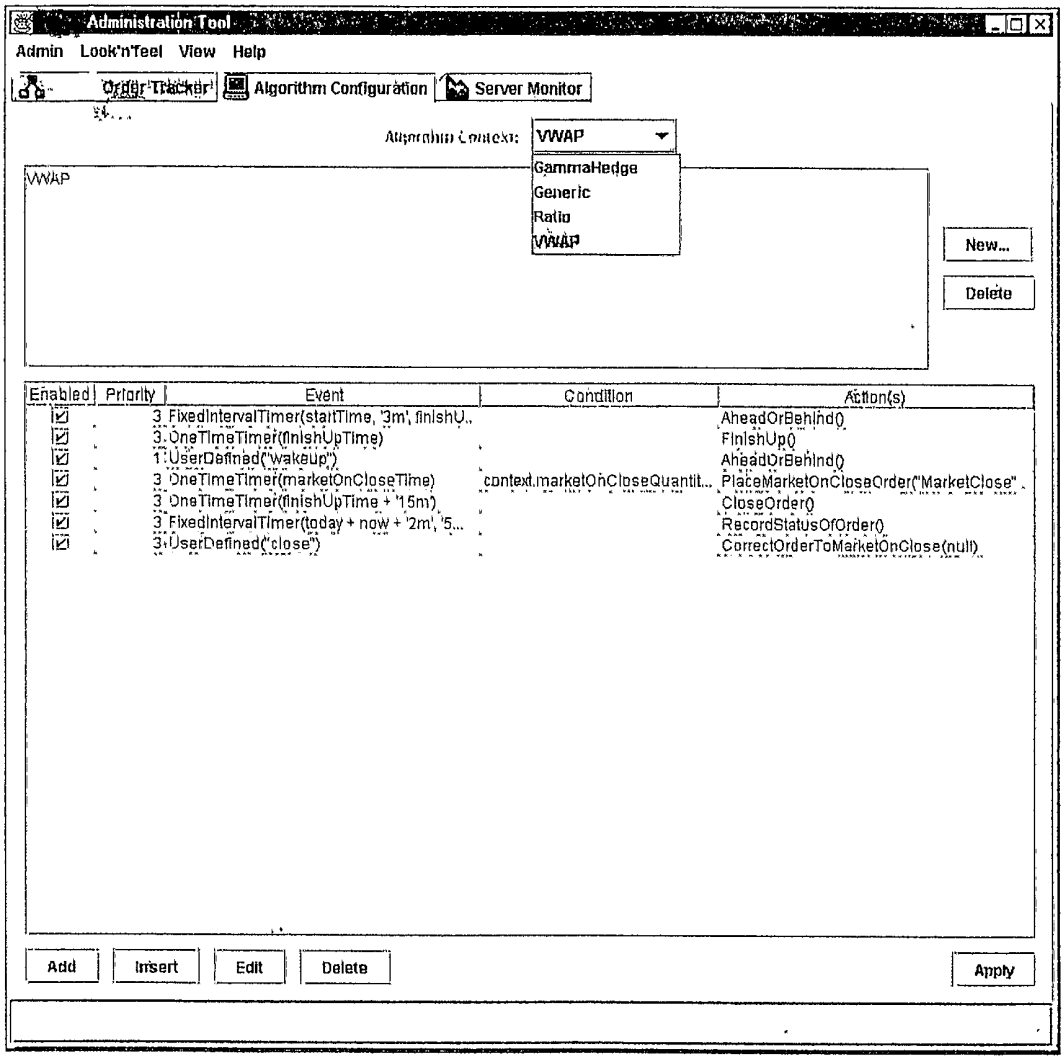

Administration Tool
Admin
Look'n'Feel
View
Help
Order Tracker
Algorithm Configuration
Server Monitor
Algorithm Context:
VWAP
GammaHedge
Generic
Ratio
VWAP
VWAP
New...
Delete
Enabled
Priority
Event
Condition
Action(s)
3 FixedIntervalTimer(startTime, '3m', finishU...
AheadOrBehind()
3 OneTimeTimer(finishUpTime)
FinishUp()
1 UserDefined("wakeup")
AheadOrBehind()
3 OneTimeTimer(marketOnCloseTime)
context.marketOnCloseQuantit...
PlaceMarketOnCloseOrder("MarketClose"
3 OneTimeTimer(finishUpTime + '15m')
CloseOrder()
3 FixedIntervalTimer(today + now + '2m', '5...
RecordStatusOfOrder()
3 UserDefined("close")
CorrectOrderToMarketOnClose(null)
Add
Insert
Edit
Delete
Apply

Figure 4

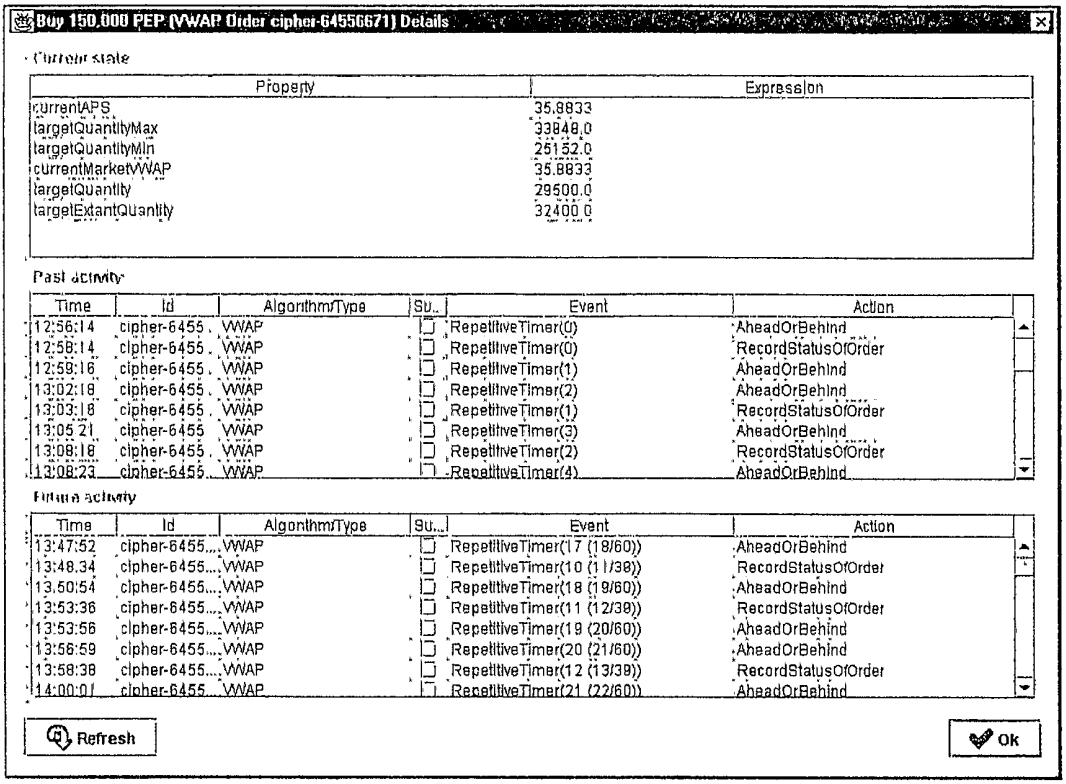

Buy 150,000 PEP (VWAP Order cipher-64556671) Details
Current state
Property
Expression
currentAPS
35.8833
targetQuantityMax
33848.0
targetQuantityMin
26152.0
currentMarketVWAP
35.8833
targetQuantity
29500.0
targetExtantQuantity
32400.0
Past activity
Time
Id
AlgorithmType
Su...
Event
Action
12:56:14 cipher-6455... VWAP RepetitiveTimer(0) AheadOrBehind
12:56:14 cipher-6455... VWAP RepetitiveTimer(0) RecordStatusOfOrder
12:56:16 cipher-6455... VWAP RepetitiveTimer(1) AheadOrBehind
13:02:18 cipher-6455... VWAP RepetitiveTimer(2) AheadOrBehind
13:03:18 cipher-6455... VWAP RepetitiveTimer(1) RecordStatusOfOrder
13:05:21 cipher-6455... VWAP RepetitiveTimer(3) AheadOrBehind
13:08:18 cipher-6455... VWAP RepetitiveTimer(2) RecordStatusOfOrder
13:08:23 cipher-6455... VWAP RepetitiveTimer(4) AheadOrBehind
Future activity
Time
Id
AlgorithmType
Su...
Event
Action
13:47:52 cipher-6455... VWAP RepetitiveTimer(17 (18/60)) AheadOrBehind
13:48:34 cipher-6455... VWAP RepetitiveTimer(10 (11/39)) RecordStatusOfOrder
13:50:54 cipher-6455... VWAP RepetitiveTimer(18 (19/60)) AheadOrBehind
13:53:36 cipher-6455... VWAP RepetitiveTimer(11 (12/39)) RecordStatusOfOrder
13:53:56 cipher-6455... VWAP RepetitiveTimer(19 (20/60)) AheadOrBehind
13:56:59 cipher-6455... VWAP RepetitiveTimer(20 (21/60)) AheadOrBehind
13:58:38 cipher-6455... VWAP RepetitiveTimer(12 (13/39)) RecordStatusOfOrder
14:00:01 cipher-6455... VWAP RepetitiveTimer(21 (22/60)) AheadOrBehind
Refresh
Ok

Figure 5

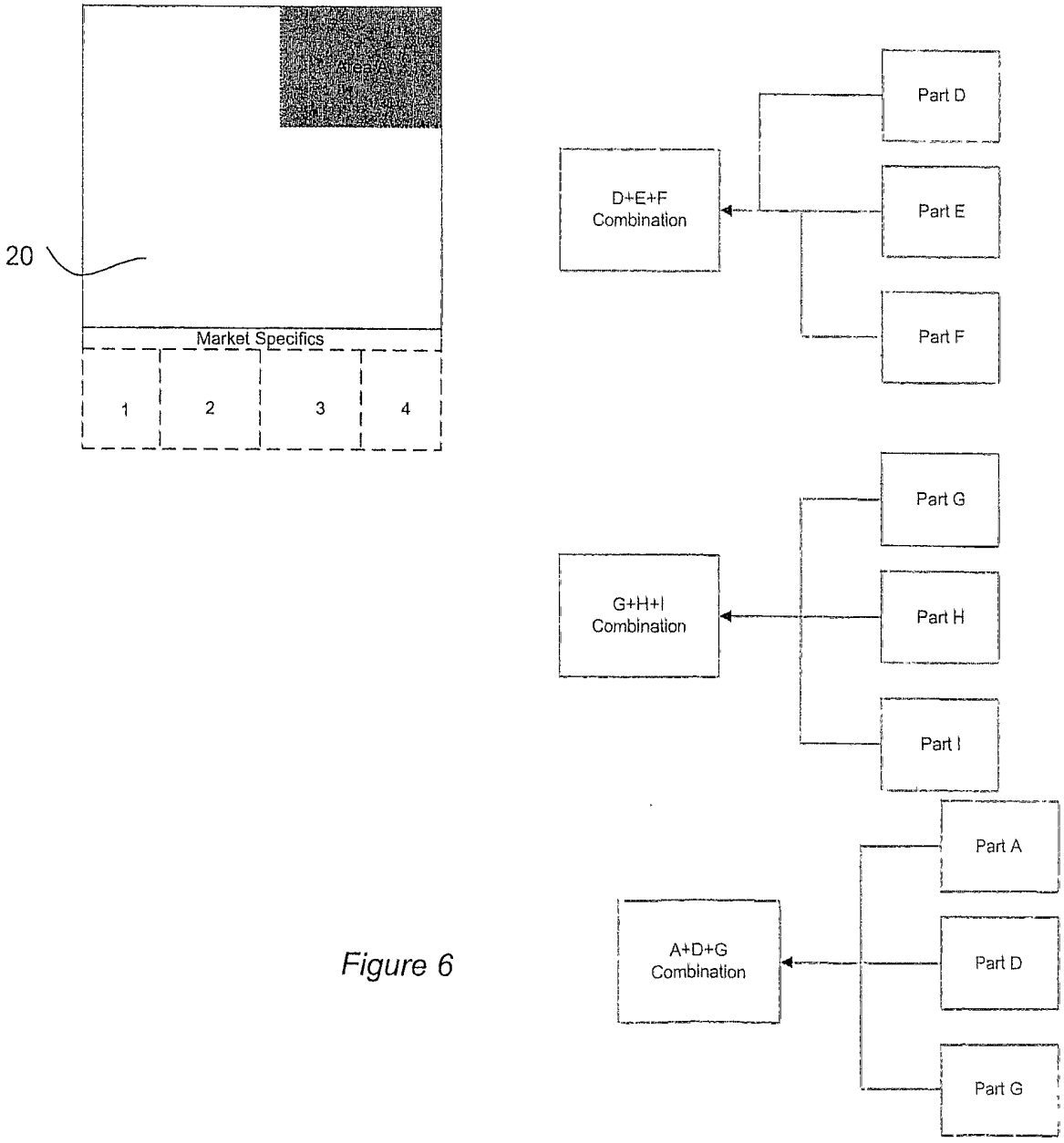

Part A
A + B + C Combination
Part B
Part C
20
Market Specifics
1
2
3
4
Part D
D+E+F Combination
Part E
Part F
Part G
G+H+I Combination
Part H
Part I
Part A
A+D+G Combination
Part D
Part G

Figure 6

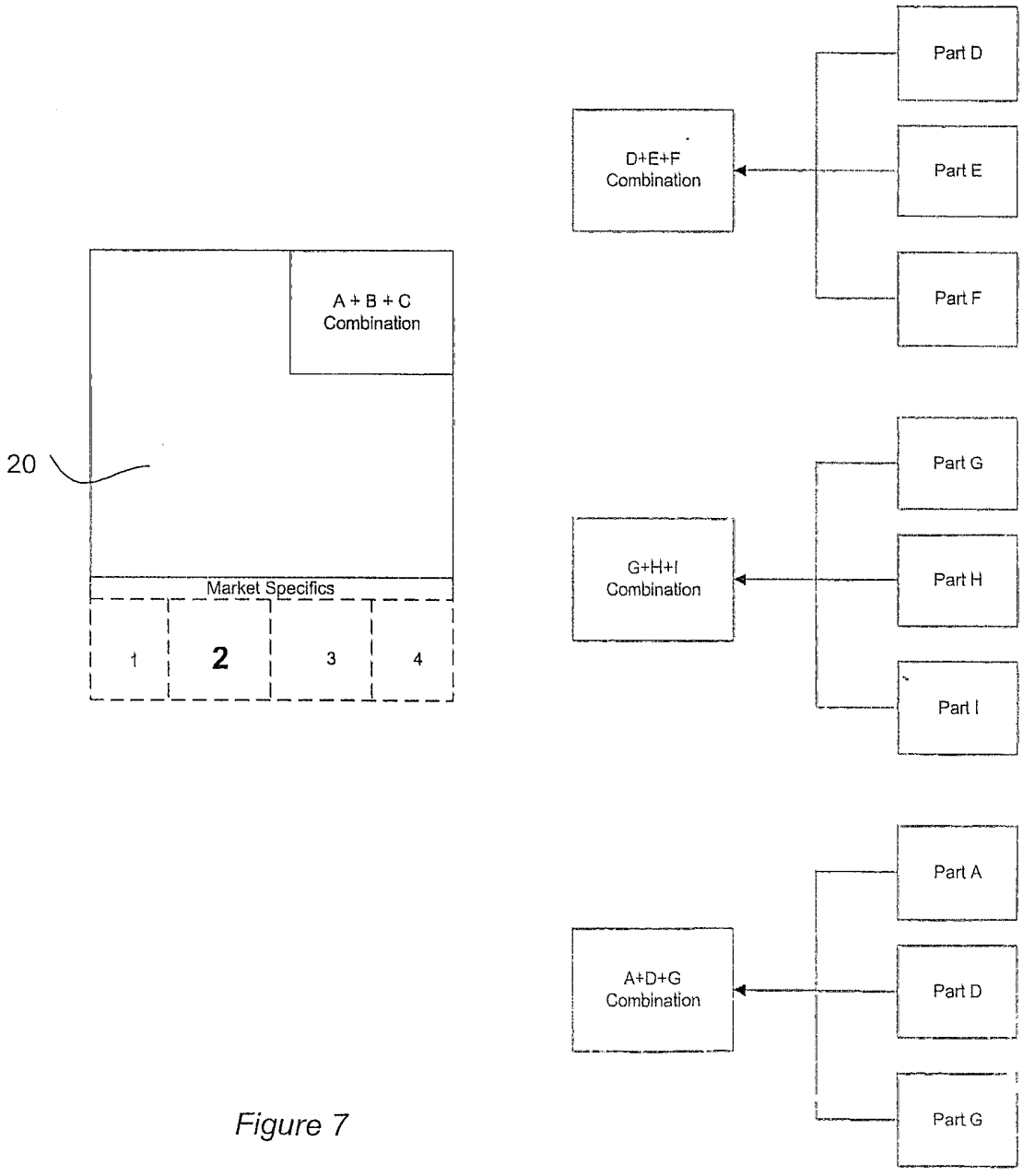

A + B + C
Combination
20
Market Specifics
1
2
3
4
D+E+F
Combination
Part D
Part E
Part F
G+H+I
Combination
Part G
Part H
Part I
A+D+G
Combination
Part A
Part D
Part G

Figure 7

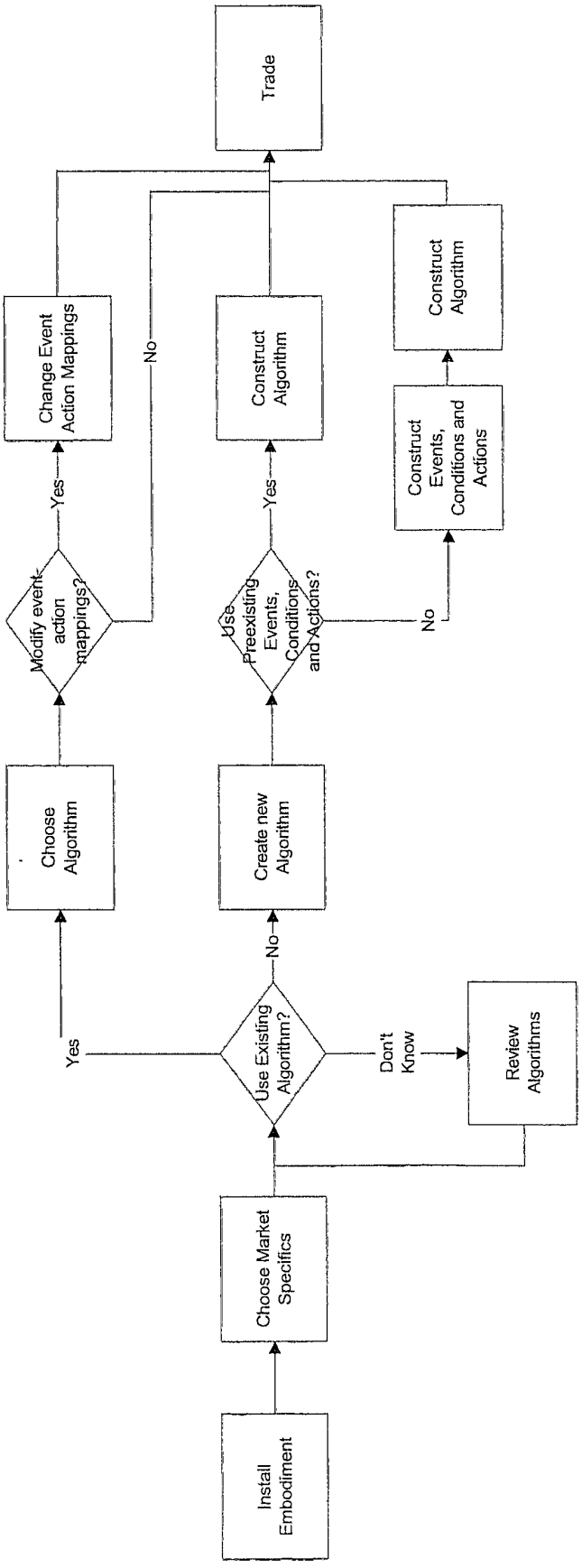

Trade
Change Event Action Mappings
Construct Algorithm
Construct Algorithm
Construct Events, Conditions and Actions
Modify event action mappings?
Use Preexisting Events, Conditions and Actions?
Yes
Yes
No
No
Choose Algorithm
Create new Algorithm
Use Existing Algorithm?
Yes
No
Don't Know
Review Algorithms
Choose Market Specifics
Install Embodiment

Figure 8

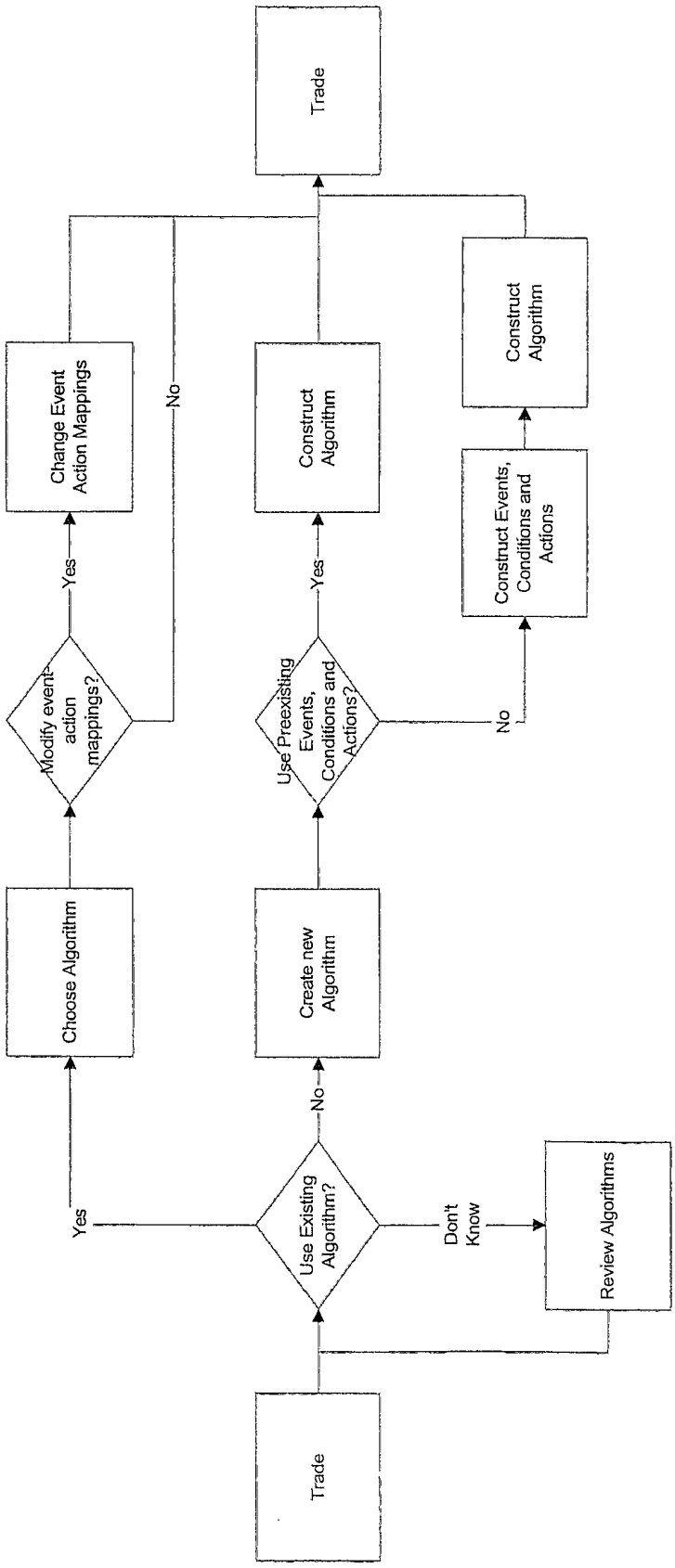

Trade
Change Event Action Mappings
Construct Algorithm
Construct Algorithm
Modify event-action mappings?
Use Preexisting Events, Conditions and Actions?
Construct Events, Conditions and Actions
Choose Algorithm
Create new Algorithm
Use Existing Algorithm?
Review Algorithms
Trade
Yes
No
Yes
No
Yes
No
Don't Know

Figure 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/42517

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 17/60
US CL : 705/37

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 705/37, 35, 36; 340/825.26, 825.27; 707/104.1

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
USPAT, JPO, EPO, Derwents WPI, IBM TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6,119,104 A (BRUMBELOW et al.) 12 September 2000 (12.09.2000), see abstract and column 4, line 61 - column 5, line 5.	1-27
Y	US 6,026,440 A (SHRADER et al.) 15 February 2000 (15.02.2000), see abstract and entire document.	1-27
A	US 5,434,395 A (STORCK et al.) 18 July 1995 (18.07.1995), see abstract.	1-27

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance, the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
23 January 2002 (23.01.2002)

Date of mailing of the international search report
14 FEB 2002

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703)305-3230

Authorized officer
Vincent Millin

Telephone No. 703 305-3900

Form PCT/ISA/210 (second sheet) (July 1998)